Jurnal Natural
Vol. 14, No. 2, 23-32, September 2014
ISSN 1141-8513

The Application of Fitness Sharing Method in Evolutionary Algorithm to Optimizing the Travelling Salesman Problem (TSP)

Nurmaulidar

Jurusan Matematika, FMIPA Universitas Syiah Kuala, Darussalam, Banda Aceh

Abstract. *Travelling Salesman Problem* (TSP) is one of complex optimization problem that is difficult to be solved, and require quite a long time for a large number of cities. Evolutionary algorithm is a precise algorithm used in solving complex optimization problem as it is part of heuristic method. Evolutionary algorithm, like many other algorithms, also experiences a premature convergence phenomenon, whereby variation is eliminated from a population of fairly fit individuals before a complete solution is achieved. Therefore it requires a method to delay the convergence. A specific method of *fitness sharing* called phenotype *fitness sharing* has been used in this research. The aim of this research is to find out whether *fitness sharing* in evolutionary algorithm is able to optimize TSP. There are two concepts of evolutionary algorithm being used in this research. the first one used single elitism and the other one used federated solution. The two concepts had been tested to the method of *fitness sharing* by using the *threshold* of 0.25, 0.50 and 0.75. The result was then compared to a non *fitness sharing* method. The result in this study indicated that by using single elitism concept, *fitness sharing* was able to give a more optimum result for the data of 100-1000 cities. On the other hand, by using federation solution concept, *fitness sharing* can yield a more optimum result for the data above 1000 cities, as well as a better solution of data-spreading compared to the method without *fitness sharing*.

Keywords: *Travelling salesman problem (TSP), evolutionary algorithm, fitness sharing.*

Pendahuluan

Travelling Salesman Problem (TSP) dikenal sebagai salah satu masalah optimasi yang banyak menarik perhatian para peneliti sejak beberapa dekade terdahulu karena permasalahan TSP termasuk ke dalam persoalan yang sulit diselesaikan dan membutuhkan waktu yang sangat lama untuk jumlah kota yang besar. TSP juga merupakan permasalahan dalam bidang diskrit dan optimasi kombinatorial. Sebagai permasalahan kombinatorial persoalan ini tergolong memiliki kemungkinan jawaban yang sangat banyak dan termasuk kedalam *Non-Polinomial Complete Problem.*

Beberapa metode telah digunakan untuk memecahkan persoalan tersebut namun hingga saat ini belum ditemukan algoritma yang tepat untuk menyelesaikannya. Cara termudah untuk menyelesaikan TSP yaitu dengan mencoba semua kemungkinan rute dan mencari rute terpendek. Namun metode tersebut memiliki kompleksitas dan waktu penyelesaian yang sangat lama jika jumlah kotanya banyak. Untuk menyelesaikan persoalan tersebut digunakan algoritma berevolusi yang merupakan bagian dari Metode Heuristik. Algoritma ini sangat tepat digunakan untuk menyelesaikan masalah optimasi kompleks yang sukar diselesaikan dengan metode konvensional [1].

Algoritma berevolusi telah diaplikasikan secara meluas pada berbagai persoalan dan telah menunjukkan performance yang baik. Meskipun demikian, algoritma berevolusi sebagaimana algoritma lainnya juga bisa mengalami fenomena kekonvergenan dini, dimana variasi dihilangkan dari suatu populasi yang terdiri dari individu-individu ideal sebelum solusi yang komplit didapatkan. Kekonvergenan dini sudah dikaji secara mendalam oleh para pakar *evolutionary*. Sejumlah mekanisme untuk pemeliharan dan peningkatan variasi di dalam suatu populasi sudah diusulkan, salah satu dari mekanisme tersebut adalah *fitness sharing* [2].

Fitness sharing yang berdasarkan pada teknik pemberian *reward* telah diusulkan untuk memelihara keanekaragaman dari suatu populasi dan mencegah kekonvergenan dini, dengan menerapkan *fitness sharing* untuk individu-individu di dalam populasi maka akan mencegah pengambilan-alih oleh individu terbaik, sehingga tidak ada individu yang muncul begitu sering dan mendominasi suatu

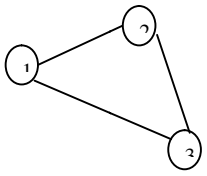
populasi. Dengan menyeimbangkan antara eksploitasi dan eksplorasi yaitu dengan menambahkan metode *fitness sharing* ke dalam algoritma berevolusi akan terlihat bahwa solusi dari *Travelling Salesman Problem* (TSP) menjadi lebih optimum.

Tinjauan Teoritis

***Travelling Salesman Problem* (TSP).** *Travelling Salesman Problem* (TSP) dideskripsikan sebagai suatu persoalan untuk menentukan urutan dari sejumlah kota yang harus dilalui oleh seorang sales. Setiap kota hanya boleh dilalui sekali dalam perjalanan dan perjalanan berakhir pada kota awal dimana seorang sales memulai perjalanannya [3].

Persoalan optimasi yang ingin dicapai adalah rute yang dilalui dan biaya yang digunakan paling minimum. Bila dipandang dari sudut komputasinya, persoalan ini dapat diselesaikan dengan cepat walaupun dengan menggunakan algoritma konvensional seperti brute force sekalipun, jika kota-kota yang akan dikunjungi sedikit. Namun, jika kota-kota yang akan dikunjungi banyak, maka algoritma konvensional tidaklah menjadi pilihan lagi karena memiliki kompleksitas dan waktu penyelesaian yang sangat lama [4].

Permasalahan pada TSP adalah mencari sirkuit terpendek pada suatu graf tidak berarah yang berasal dari suatu simpul dengan melewati seluruh simpul dan kembali ke simpul asal. Dalam menyelesaikan permasalahan TSP, kota dapat dinyatakan sebagai sebuah simpul graf, sedangkan sisi menyatakan jalan yang menghubungkan antara dua kota dan bobot pada sisi menyatakan jarak antara dua kota [5]. Jika setiap simpul pada graf berbobot mempunyai sisi ke simpul lain maka graf tersebut adalah graf lengkap berbobot. Pada sembarang graf lengkap dengan n buah simpul untuk $n > 2$ maka jumlah sirkuit hamilton yang berbeda adalah $\frac{(n-1)!}{2}$.
Berikut adalah beberapa contoh penangganan permasalahan *Travelling Salesman Problem* dengan menggunakan sirkuit hamilton:

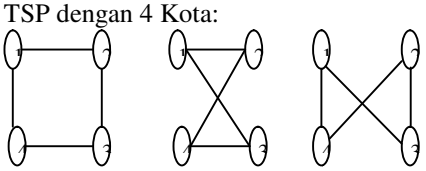


1. TSP dengan 3 kota:

Gambar 1. TSP dengan 3 kota

Permasalahan TSP dengan 3 kota tidak memerlukan komputasi karena jumlah kemungkinan solusi hanya satu.

TSP dengan 4 Kota:



Gambar 2. TSP dengan 4 kota

Permasalahan TSP dengan 4 kota belum memerlukan komputasi karena jumlah kemungkinan solusinya hanya (4-1)!/2 = 3.

2. TSP dengan 5 kota:
   Permasalahan dengan 5 kota mulai memerlukan komputasi karena jumlah kemungkinan solusi adalah (5-1)!/2=12.

Permasalahan dengan jumlah kota yang semakin banyak akan menghasilkan semakin banyak kemungkinan. Untuk 20 kota akan menghasilkan 19!/2 atau sekitar 6,08 x 10^{16} kemungkinan solusi. Sampai saat ini, belum ada algoritma yang dapat menghasilkan solusi terbaik dengan waktu yang lebih cepat untuk persoalan TSP. Algoritma yang ditemukan hingga saat ini belum dapat melakukan komputasi dengan banyak input untuk menentukan solusi TSP. Hal yang dapat dilakukan adalah mencari solusi paling optimal yaitu solusi yang memiliki kesesuaian antara waktu pemrosesan dengan hasil yang didapatkan.

Algoritma Berevolusi. Algoritma berevolusi adalah algoritma yang dikembangkan dari proses pencarian solusi optimasi menggunakan pencarian acak, ini terlihat dari proses pembangkitan populasi awal yang menyatakan sekumpulan solusi yang terpilih secara acak. Algoritma ini memanfaatkan proses seleksi alamiah yang dikenal dengan proses evolusi. Dalam proses evolusi, individu secara terus-menerus mengalami perubahan gen untuk menyesuaikan diri dengan lingkungan hidupnya. Hanya individu yang kuat yang mampu bertahan, sehingga dalam proses evolusi diharapkan dapat memperoleh individu yang terbaik [6].

Algoritma berevolusi digunakan untuk mendapatkan solusi yang tepat dari masalah optimasi dengan satu variabel atau multi variabel. Berbeda dengan teknik pencarian konvensional, algoritma berevolusi bermula dari himpunan solusi yang dihasilkan secara acak. Himpunan ini disebut populasi, sedangkan setiap individu didalam populasi disebut kromosom yang merupakan representasi dari solusi. Kromosom-kromosom berevolusi dalam suatu proses iterasi yang berkelanjutan yang disebut generasi. Pada setiap generasi, kromosom dievaluasi

berdasarkan suatu fungsi evaluasi sampai didapatkan suatu individu yang dianggap optimal [7].

Fitness Sharing. Gagasan utama dari *fitness sharing* adalah untuk membagi-bagi suatu populasi individu berdasarkan letaknya didalam *search space*. Ketika suatu individu *i* mempunyai letak yang berdekatan dengan individu lain disekelilingnya, *fitness* f_i diturunkan secara proporsional terhadap jumlah kedekatan tersebut. *Fitness sharing* juga memodifikasi peningkatan pencarian dengan mengurangi *reward* untuk daerah yang padat populasi. *Fitness sharing* menurunkan *fitness* dari tiap-tiap anggota populasi dengan jumlah yang hampir sama dengan banyaknya individu yang memiliki kemiripan didalam populasi. *Fitness* f_i yang dibagi dari suatu individu i dengan *fitness* f_i secara sederhana dapat dituliskan:

$$f_i' = \frac{f_i}{m_i}$$

Dimana m_i adalah hitungan niche (kedudukan yang sesuai) untuk mengukur jumlah jarak yang berdekatan dari individu-individu dengan siapa *fitness* f_i dibagi. Hitungan niche dihitung dengan menjumlahkan suatu fungsi *sharing* atas semua anggota populasi. Secara matematis m_i dapat dituliskan seperti berikut ini:

$$mi = \sum_{j=1}^{N} Sh\left(d_{ij}\right)$$

Dimana N merupakan ukuran populasi dan d_{ij} adalah jarak antara individu i dan individu j, kemudian fungsi *sharing* (Sh) mengukur tingkat kesamaan antara dua anggota populasi. Nilainya 1 jika individu-individu tersebut identik, bernilai 0 jika jarak d_{ij} lebih tinggi dari batas *dissimilarity* dan suatu nilai tengah pada perbedaan jarak tingkat menengah. Secara meluas penggunaan fungsi *sharing* diberikan seperti dibawah ini:

$$Sh\left(d_{ij}\right) = \begin{cases} 1-(\frac{d_{ij}}{\sigma_s})^{\alpha}, & if\ d<\sigma_s \\ 0, & otherwise \end{cases}$$

Dimana σ_s merupakan batas *dissimilarity* (*threshold*) dan α adalah parameter konstanta yang mengatur bentuk dari fungsi *sharing*. Nilai *threshold* dideskripsikan sebagai batas toleransi untuk individu-individu di dalam populasi. Jika jarak antar individu-individu tersebut lebih kecil dari batas toleransi, maka individu-individu tersebut akan diberikan pinalti. Sebaliknya, jika jaraknya lebih besar dari *threshold,* maka individu-individu tersebut terbebas dari pinalti. Semakin besar nilai *threshold,* maka semakin banyak individu-individu di dalam populasi yang akan mendapatkan pinalti, Sehingga, dengan *threshold* yang besar diharapkan akan semakin besar ketidakmiripan antara individu-individu di dalam populasi atau semakin jauh letak individu tersebut dengan individu lain di sekililingnya. Nilai untuk suatu individu i adalah $0 < f_i \leq 1$. Jarak antara individu i dan j mempunyai range $0 \leq d_{ij} < 1$, sedangkan nilai *threshold* mempunyai batasan $0 < \sigma_s \leq 1$.

Parameter α adalah suatu nilai yang berbentuk bilangan bulat atau integer. Nilai α yang digunakan dalam penelitian ini adalah 1. Hal ini dikarenakan $\alpha = 1$ adalah nilai yang sangat sering digunakan dalam penelitian untuk kasus optimasi seperti TSP. Tidak ada informasi yang lebih jauh tentang penentuan nilai α yang paling sesuai untuk kasus-kasus optimasi seperti ini. Namun, berdasarkan rumus di atas dapat dianalisis bahwa jika nilai α semakin besar atau $\alpha \to \infty$ maka $\frac{d_{ij}}{\sigma_s} \to 0$, $Sh\left(d_{ij}\right) \to 1$ dengan batasan $0 \leq Sh\left(d_{ij}\right) \leq 1$, karena $Sh\left(d_{ij}\right) \to 1$ berarti $Sh\left(d_{ij}\right)$ menuju ke suatu nilai yang besar, ini menyebabkan nilai m_i menjadi besar, sehingga f_i' cendrung untuk mempunyai nilai yang kecil. Akan tetapi, jika $\alpha \to -\infty$, $\frac{d_{ij}}{\sigma_s} \to 0$, $Sh\left(d_{ij}\right)$ akan menghasilkan nilai yang negatif, menyebabkan nilai f_i' juga bernilai negatif. Kenyataan ini menjadi kontradiksi dengan pernyataan bahwa nilai f_i' selalu berada diantara 0 dan 1. Begitu juga jika nilai $\alpha = 0$, akan menghasilkan $\frac{d_{ij}}{\sigma_s} = 1$, sehingga $Sh\left(d_{ij}\right) = 0$, mengakibatkan $m_i = 0$. Hal ini tidak akan mempunyai arti apapun karena menyebabkan nilai f_i' tidak terdefinisi. Dengan demikian, nilai α harus selalu lebih besar dari 0, dengan konsekuensi jika α semakin jauh dari 1 atau $\alpha \to \infty$, maka f_i' akan cendrung untuk mempunyai nilai yang kecil. Oleh karena itulah, nilai α yang sering digunakan di sini adalah $\alpha = 1$.

Metode Penelitian

Studi Literatur

Studi literatur mengenai *Traveling Salesman Problem* (TSP). Algoritma berevolusi dan *fitness sharing*.

Desain Algoritma dan Pembuatan Program Komputer

1. Menyusun *flowchart* algoritma berevolusi.
2. Membuat program berevolusi untuk memecahkan permasalahan TSP. Pemrograman akan dilakukan dalam bahasa matlab. Tahapan dalam melakukan pemrograman adalah:
   a. Membaca file data TSP.
   b. Inisialisasi: menentukan ukuran populasi, jumlah generasi, banyaknya *offspring*, *treshold*, gen dan kromosom. Jumlah gen dalam setiap kromosom sama dengan jumlah kota.
   c. Evaluasi kromosom: melakukan evaluasi fungsi objektif yaitu dengan mencari nilai *cost function* yang lebih minimum dengan menggunakan rumus jarak kartesian antara kota A dan kota B:

   $$\|A-B\|=(\sqrt{(X_A-X_B)^2+(Y_A-Y_B)^2}$$

   dengan (X_A, Y_A) menyatakan posisi koordinat kota A dan (X_B, Y_B) adalah posisi koordinat kota B. Kemudian fungsi objektif dikonversikan kedalam fungsi *fitness.*

   d. Seleksi kromosom: metode seleksi kromosom yang digunakan adalah metode *roulette wheel*, dimana masing-masing kromosom dengan nilai *fitness* tinggi menempati potongan lingkaran yang lebih besar dibandingkan dengan kromosom yang bernilai rendah. Program *roulette wheel* yang digunakan di *download* dari GAOT (*Genetic Algorithm Optimization Toolbox*).
   e. *Fitness sharing*: dalam melakukan *fitness sharing*, akan dihitung fungsi *sharing* terlebih dahulu dengan menggunakan rumus berikut:

   $$Sh(d_{ij}) = \begin{cases} 1-(\frac{d_{ij}}{\sigma_s})^{\alpha}, & if\ d<\sigma_s \\ 0, & otherwise \end{cases}$$

   Dengan d_{ij} adalah jarak antara individu i dan j, dalam hal ini individu adalah urutan antar kota yang direpresentasikan sebagai satu solusi yang valid dan σ_s sebagai *treshold*. Setelah fungsi *sharing* didapat maka langkah selanjutnya adalah menghitung nilai pinalti untuk masing-masing individu dengan menggunakan rumus:

   $$m_i = \sum_{j=1}^{N} Sh\,(d_{ij})$$

   Selanjutnya adalah mencari nilai *fitness sharing* dari individu-individu yang sudah dishare dengan menggunakan rumus di bawah ini:

   $$f_i' = \frac{f_i}{m_i}$$

   Dengan f_i adalah fungsi *fitness* dan f_i' adalah *fitness sharing* dari individu i.
   d. *Crossover* dan mutasi: metode *crossover* yang digunakan adalah *order-based crossover* dan metode mutasi yang digunakan adalah *swap mutation.* Program *order-based crossover* dan *swap mutation* di *download* dari GAOT.
   e. *Elitism*: dalam setiap generasi dilakukan penyalinan beberapa kromosom terbaik ke dalam populasi baru. Dalam penelitian ini digunakan konsep dengan 1 elitism dan n elitism.
   f. Populasi baru: dari proses *fitness sharing*, *crossover*, mutasi dan *elitism* terbentuklah populasi baru dengan nilai *fitness* yang lebih baik.

Pengujian Komputasi. Data yang digunakan dalam pengujian komputasi adalah data sekunder yang didownload dari: http://www.iwr.uniheidelberg.de/groups/comopt/software/TSPLIB95. Spesifik data TSP yang digunakan dapat dilihat pada Tabel1.

Tabel 1. Dataset TSP dengan jumlah kota (100-1500)

No	Data	Jumlah Kota	*Edge Type*
1	kroA100	100	EUC_2D
2	kroA150	150	EUC_2D
3	kroB150	150	EUC_2D
4	u159	159	EUC_2D
5	gil262	262	EUC_2D
6	fl417	417	EUC_2D
7	d493	493	EUC_2D
8	att532	532	EUC_2D
9	rat575	575	EUC_2D
10	p654	654	EUC_2D
11	d657	657	EUC_2D
12	rat783	783	EUC_2D
13	u817	817	EUC_2D
14	rl934	934	EUC_2D
15	pr1002	1002	EUC_2D
16	pcb1173	1173	EUC_2D
17	d1291	1291	EUC_2D
18	rl1323	1323	EUC_2D
19	fl1400	1400	EUC_2D
20	u1432	1432	EUC_2D
21	fl1577	1577	EUC_2D

Pengujian Komputasi untuk data di atas dilakukan dengan menggunakan parameter-parameter pengujian. Parameter yang digunakan untuk maksimum generasi adalah 1000, *population size* adalah 10, nilai faktor pembanding *offspring* 5 dan nilai *treshold* yang digunakan adalah 0.25, 0.50, dan 0.75. Pengujian komputasi dilakukan untuk melihat nilai minimum solusi, populasi solusi dan standar deviasi dari metode yang menggunakan *fitness sharing* dengan metode yang tanpa *fitness sharing*. Konsep Algoritma berevolusi yang digunakan adalah dengan 1 elitism dan n elitism. Simulasi ini dilakukan sebanyak 10 kali percobaan untuk masing-masing dataset, untuk masing-masing konsep dan untuk setiap metode. Hasil yang diperoleh kemudian diolah dengan menggunakan *software Excel* untuk mencari nilai rata-rata dari masing-masing data. Tahap selanjutnya adalah membandingkan hasil yang didapat dari metode *fitness sharing* dengan metode *non fitness sharing.*

Hasil dan Pembahasan

Simulasi dan Hasil. Dalam pembahasan ini digunakan 2 konsep dari Algoritma berevolusi, konsep pertama dengan 1 elitism dan konsep kedua dengan n elitism. Untuk setiap konsep, simulasi dilakukan terhadap metode algoritma berevolusi. Algoritma berevolusi ini diuji pada *dataset* TSPLIB95 dengan jumlah kota antara 100 sampai 1500 kota. Nilai *treshold* (T) yang digunakan dikatagorikan menjadi 3 yaitu: *treshold* berukuran kecil 0.25, ukuran sedang 0.50 dan ukuran besar 0.75. Hasil simulasi dari solusi TSP yang menggunakan *fitness sharing* (FS) secara langsung dibandingkan dengan algoritma yang tanpa *fitness sharing* (Non FS). Untuk memudahkan dalam melakukan analisa, data dipisahkan ke dalam 3 bagian yaitu data ukuran kecil yang terdiri dari 100-500 kota, data ukuran sedang terdiri dari 500-1000 kota dan data ukuran besar dengan jumlah kota lebih dari 1000 kota. Untuk setiap *dataset* dilakukan 10 kali percobaaan. Hasil percobaan dari setiap *dataset* diperlihatkan seperti berikut:

Konsep dengan 1 elitism. Hasil simulasi untuk metode *fitness sharing* dan *non-fitness sharing* untuk solusi minimum dan nilai standar deviasi yang menggunakan 1 elitism akan diperlihatkan pada tabel-tabel berikut ini. Tabel 2 memperlihatkan perbandingan nilai solusi minimum yang dihasilkan dari metode *fitness sharing* dengan metode *non fitness sharing*. Untuk data ukuran kecil rata-rata nilai yang dihasilkan dengan metode *fitness sharing* lebih kecil sebesar 0.20% dan 0.42% dengan *treshold* yang berturut-turut 0.25 dan 0.75. Akan tetapi dengan *treshold* 0.50 rata-rata nilai yang dihasilkan lebih maksimum sebesar 0.16%. Tabel 3 memperlihatkan bahwa nilai solusi minimum untuk data ukuran sedang dengan metode *fitness sharing* menghasilkan nilai yang lebih kecil sebesar 0.11%, 0.45% dan 0.43% untuk *treshold* yang berturut-turut 0.25, 0.50 dan 0.75.

Tabel 2. Nilai solusi minimum untuk data ukuran kecil

Data	Kota	Non FS	*Fitness sharing*					
			T(0.25)	%	T(0.50)	%	T(0.75)	%
KroA100	100	135,103	136,845	1.29%	137,633	1.87%	136,101	0.74%
KroA150	150	216,853	214,691	-1.00%	215,538	-0.61%	214,189	-1.23%
KroB150	150	213,531	213,312	-0.10%	213,341	-0.09%	213,295	-0.11%
u159	159	381,379	380,155	-0.32%	377,590	-0.99%	378,322	-0.80%
gil262	262	23,647	23,621	-0.11%	23,820	0.73%	23,837	0.80%
fl417	417	441,306	434,830	-1.47%	439,094	-0.50%	429,370	-2.70%
d493	493	405,842	407,027	0.29%	408,777	0.72%	407,214	0.34%
Average			**-0.20%**		**0.16%**		**-0.42%**	

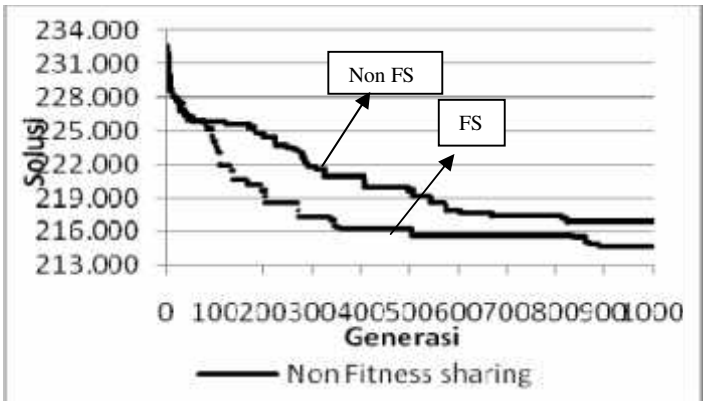
Tabel 3. Nilai solusi minimum untuk data ukuran sedang

Data	Kota	Non FS	*Fitness sharing*					
			T(0.25)	%	T(0.50)	%	T(0.75)	%
att532	532	1,474,094	1,460,325	-0.93%	1,449,756	-1.65%	1,446,141	-1.90%
rat575	575	103,253	103,950	0.68%	103,276	0.02%	102,404	-0.82%
p654	654	1,839,741	1,826,907	-0.70%	1,837,955	-0.10%	1,828,430	-0.61%
d657	657	789,455	792,171	0.34%	783,435	-0.76%	791,484	0.26%
rat783	783	165,508	165,723	0.13%	165,749	0.15%	164,845	-0.40%
u817	817	939629	938556	-0.11%	931145.3	-0.90%	940399	0.08%
rl934	934	6314101	6301598	-0.20%	6321091	0.11%	6E+06	0.39%
Average			**-0.11%**		**-0.45%**		**-0.43%**	

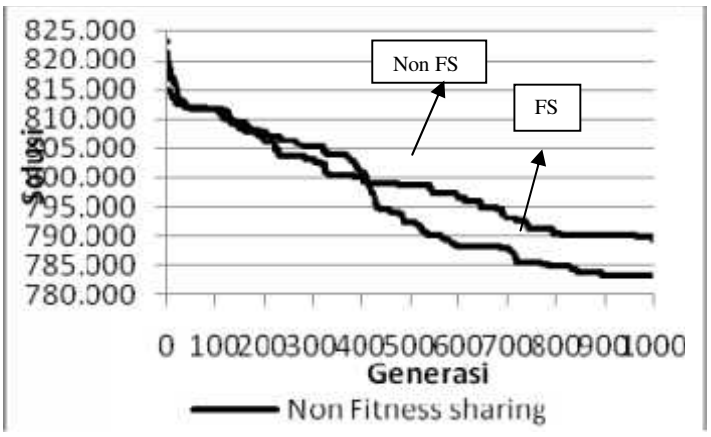
Tabel 4. Nilai solusi minimum untuk data ukuran besar

Data	Kota	Non FS	*Fitness sharing*					
			T(0.25)	%	T(0.50)	%	T(0.75)	%
PR1002	1002	6,052,365	6,045,429	-0.11%	6,047,893	-0.07%	6,039,557	-0.21%
pcb1173	1173	159,202	158,135	-0.67%	162,322	1.96%	159,368	0.10%
d1291	1291	1,637,558	1,640,244	0.16%	1,634,159	-0.21%	1,637,049	-0.03%
rl1323	1323	9,248,501	9,218,792	-0.32%	9,247,956	-0.01%	9,230,617	-0.19%
fl1400	1400	1,570,858	1,560,756	-0.64%	1,569,135	-0.11%	1,572,696	0.12%
u1432	1432	3,703,269	3,727,861	0.66%	3,730,550	0.74%	3,764,869	1.66%
fl1577	1577	1,279,027	1,283,317	0.34%	1,294,369	1.20%	1,289,608	0.83%
Average			**-0.08%**		**0.50%**		**0.33%**	

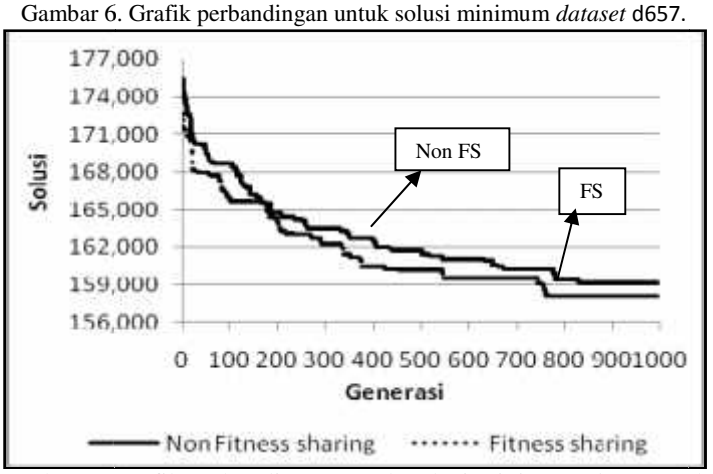
Untuk data ukuran besar seperti yang diperlihatkan pada tabel 4, nilai solusi minimum dengan metode *fitness sharing* menghasilkan rata-rata nilai yang lebih kecil 0.08% dengan *treshold* 0.25 dan nilainya lebih maksimum 0.50% dan 0.33% dengan *treshold* yang berturut-turut 0.50 dan 0.75.Perbandingan nilai dari metode *fitness sharing* dengan metode *non fitness sharing* untuk 1 elitism dengan mengambil nilai rata-rata dari 10 kali percobaan untuk setiap *dataset* juga dapat dilihat dalam bentuk grafik 5,6 dan 7 di bawah ini. Grafik yang diperlihatkan hanya diambil 3 sampel dari setiap kelompok data, dimulai dari data ukuran kecil, sedang dan besar. Sumbu x menyatakan banyaknya generasi dan sumbu y menyatakan solusi dari TSP.



Gambar 5. Grafik perbandingan untuk solusi minimum *dataset* kroA150



Gambar 6. Grafik perbandingan untuk solusi minimum *dataset* d657.



Gambar 7. Grafik perbandingan untuk solusi minimum *dataset* pcb1173

Konsep dengann elitism (solusi gabungan). Tabel 5, 6 dan 7 memperlihatkan bahwa nilai solusi minimum untuk data ukuran kecil dan sedang dengan n elitism dengan menggunakan *fitness sharing* memberikan hasil yang lebih maksimum. Untuk data ukuran besar dengan *treshold* 0.25 hasil yang didapatkan menjadi lebih maksimum, akan tetapi dengan menggunakan *treshold* 0.50 dan 0.75 hasilnya menjadi lebih minimum sebesar 0.58% dan 0.21%. Perbandingan nilai dari metode *fitness sharing* dengan metode *non fitness sharing* yang menggunakan n elitism juga dapat dilihat dalam bentuk grafik di bawah ini. Grafik yang diperlihatkan hanya diambil 3 sampel dari setiap kelompok data, dimulai dari ukuran kecil, sedang dan besar. Sumbu x menyatakan banyaknya generasi dan sumbu y menyatakan solusi dari TSP.

Tabel 5. Nilai solusi minimum untuk data ukuran kecil dengan n elitism

Data	**Kota**	**Non FS**	***Fitness sharing***					
			FS(0.25)	**%**	**FS(0.50)**	**%**	**FS(0.75)**	**%**
KroA100	100	97679	99968	2.34%	103509	5.97%	98554	0.90%
KroA150	150	165139	167300	1.31%	157063	-4.89%	156178	-5.43%
KroB150	150	159180	155413	-2.37%	163103	2.46%	160076	0.56%
u159	159	280079	289873	3.50%	289324	3.30%	282726	0.95%
gil262	262	18075	18611	2.97%	18555	2.65%	19017	5.21%
fl417	417	321293	324580	1.02%	314148	-2.22%	321192	-0.03%
d493	493	316491	307603	-2.81%	312007	-1.42%	309950	-2.07%
Average			**0.85%**		**0.84%**		**0.01%**	

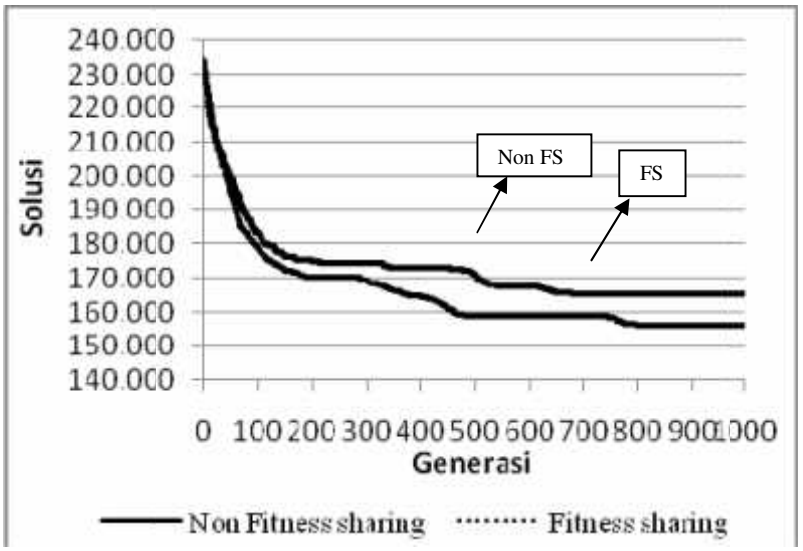
Tabel 6. Nilai solusi minimum untuk data ukuran sedang dengan n elitism

Data	**Kota**	**Non FS**	***Fitness sharing***					
			FS(0.25)	**%**	**FS(0.50)**	**%**	**FS(0.75)**	**%**
att532	532	1070774	1122594	4.84%	1106463	3.33%	1078178	0.69%
rat575	575	79807	80749	1.18%	80909	1.38%	80154	0.43%
p654	654	1297964	1295529	-0.19%	1309095	0.86%	1326582	2.20%
d657	657	628040	619047	-1.43%	642659	2.33%	621441	-1.05%
rat783	783	127110	130910	2.99%	128822	1.35%	128683	1.24%
u817	817	689396	698832	1.37%	684385	-0.73%	696839	1.08%
rl934	934	4929305	4979944	1.03%	4947623	0.37%	5125259	3.98%

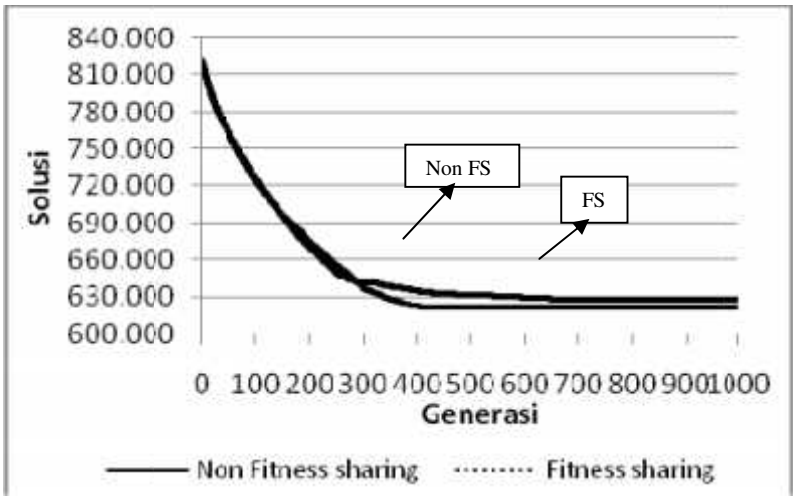
Average		**1.46%**	**1.42%**	**0.77%**

Tabel 7. Nilai solusi minimum untuk data ukuran besar dengan n elitism

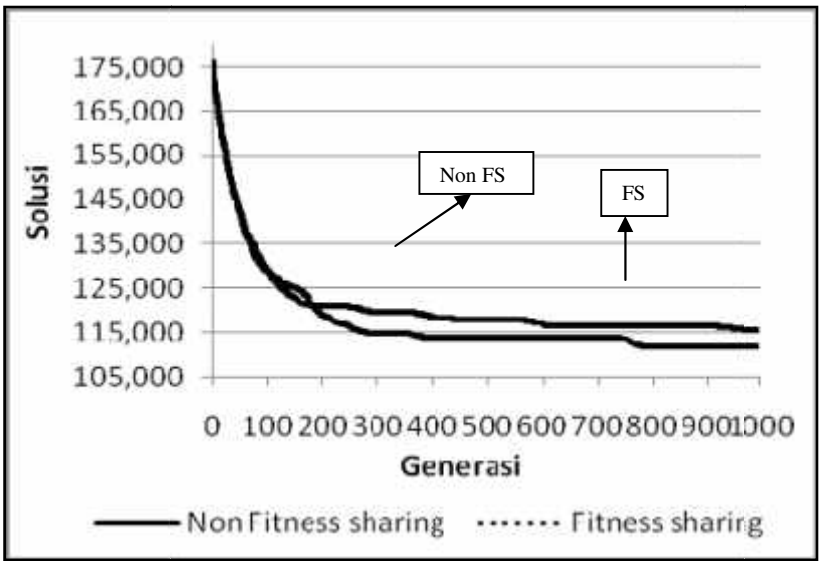
Data	**Kota**	**Non FS**	***Fitness sharing***					
			FS(0.25)	**%**	**FS(0.50)**	**%**	**FS(0.75)**	**%**
PR1002	1002	4775673	4702091	-1.54%	4773503	-0.05%	4769920	-0.12%
pcb1173	1173	115547	116230	0.59%	111804	-3.24%	115980	0.37%
d1291	1291	1298813	1320376	1.66%	1295588	-0.25%	1307242	0.65%
rl1323	1323	7350830	7375619	0.34%	7356694	0.08%	7301440	-0.67%
fl1400	1400	1146842	1152755	0.52%	1153708	0.60%	1114666	-2.81%
u1432	1432	2942520	2990466	1.63%	2929744	-0.43%	2957036	0.49%
fl1577	1577	1017496	1007738	-0.96%	1009348	-0.80%	1023883	0.63%
Average			**0.32%**		**-0.58%**		**-0.21%**	



Gambar 8. Grafik perbandingan untuk solusi minimum *dataset* kroA150.



Gamba 9. Grafik perbandingan untuk solusi minimum *dataset* d657.



Gambar 10.Grafik perbandingan untuk solusi minimum *dataset* pcb1173.

Pembahasan

Hasil yang diperlihatkan pada tabel 3-4 dan tabel 5-7 dapat diringkas ke dalam tabel 8. Pada tabel 8 terlihat bahwa *fitness sharing* dengan semua ukuran *treshold* dapat mempengaruhi kualitas solusi (nilai solusi minimum dan solusi populasi) dengan 1 elitism untuk data ukuran kecil dan sedang, sedangkan untuk data ukuran besar hasilnya lebih minimum dengan menggunakan *treshold* yang kecil. Hal ini dikarenakan *treshold* yang kecil dapat meminimalkan pergerakan acak yang ditimbulkan oleh sistem *replacing* dengan 1 elitism. Pergerakan acak yang kuat juga disebabkan oleh ukuran *search space* yang besar dan *treshold* yang besar. Sistem *replacing* dengan 1 elitism dapat membuat solusi menjadi menyebar di dalam *search space*, akan tetapi penyebarannya tidak memberikan pengaruh yang bagus terhadap kualitas solusi TSP.

Tabel 8. Pengaruh *fitness sharing* terhadap kualitas solusi TSP yang menggunakan 1 elitism dan n elitism

Ukuran data	**Nilai Solusi**	
	1 elitism	**n elitism**
Kecil	Pengaruh (dengan semua *treshold*)	Tidak pengaruh
Sedang	Pengaruh (dengan semua *treshold*)	Tidak pengaruh
Besar	Pengaruh (dengan *treshold* yang kecil)	Pengaruh (dengan *treshold* yang besar)

Ini dikarenakan penyebaran dengan 1 elitism adalah penyebaran acak yang tanpa kontrol, sedangkan penyebaran dengan *fitness sharing* dilakukan secara terkontrol dengan ukuran *treshold* tertentu. Oleh karena itu, penyebaran dengan *fitness sharing* dapat mempengaruhi kualitas solusi TSP menjadi lebih baik. Tabel 8 juga memperlihatkan pengaruh *fitness sharing* terhadap kualitas solusi TSP dengan n elitism (solusi gabungan). Untuk data ukuran kecil dan sedang, *fitness sharing* tidak memberikan hasil yang lebih bagus. Hal ini dikarenakan sistem *replacing* setengah acak sudah sangat efisien untuk *search space* ukuran kecil dan sedang. Ukuran *search space* yang kecil juga menyebabkan kemungkinan terjadinya

minimum lokal menjadi lebih kecil, sehingga penambahan *fitness sharing* tidak memberikan pengaruh yang signifikan. Sementara, untuk data ukuran besar hasilnya lebih minimum dengan menggunakan *treshold* yang besar. Ini dikarenakan *treshold* yang besar mampu melakukan penyebaran solusi yang lebih luas, sehingga dapat menjangkau keseluruh sisi *space* ukuran besar. Dengan demikian, probabilitas untuk menemukan solusi-solusi baru yang optimum menjadi lebih besar.

Kesimpulan

Berdasarkan pembahasan di atas dapat diambil kesimpulan sebagai berikut:

1. *Fitness sharing* dengan semua *treshold* dapat meminimumkan solusi TSP yang menggunakan 1 elitism untuk data ukuran kecil dan sedang. Untuk data ukuran besar solusi TSP menjadi lebih minimum dengan menggunakan *treshold* yang kecil.
2. *Fitness sharing* dapat memberikan hasil yang lebih bagus untuk solusi TSP dengan 1 elitism.
3. *Fitness sharing* dengan semua *treshold* tidak mempengaruhi kualitas solusi TSP yang menggunakan n elitism untuk data ukuran kecil dan sedang. Untuk data ukuran besar, *fitness sharing* dengan *treshold* yang besar dapat mempengaruhi kualitas solusi TSP menjadi lebih baik.
4. *Fitness sharing* dapat mempengaruhi kualitas solusi TSP yang menggunakan n elitism menjadi lebih baik untuk data ukuran besar.
5. *Fitness sharing* dalam algoritma berevolusi dapat mengoptimumkan kualitas solusi TSP baik yang menggunakan 1 elitism maupun n elitism.

Daftar Pustaka

1. B. Ahmad. 2003. *Algoritma Genetika Suatu Alternatif Penyelesaian Permasalahan Searching, Optimasi dan Machine Learning*, Politeknik Elektronika Negeri Surabaya, Surabaya.

2. F. M. Irvan. 2007. *Perbandingan Penggunaan Algoritma Genetika dengan Algoritma Konvensional pada Traveling Salesman Problem*, ITB, Bandung.

3. F. Filman. 2009. *Penyelesaian Travelling salesman problem dengan Algoritma heuristic*, http://mail.informatika.org/~rinaldi/Matdis/20082009/Makalah2008/Makalah0809-028.pdf, diakses pada tanggal 28 Desember 2009.

4. F. Aulia. 2006. *Penerapan Algoritma Genetika pada persoalan pedagang Keliling,* ITB, Bandung. .

5. G. Mitsuo, C. Runwei. 1997. *Genetic Algorithm and Engineering Design,* John Wiley & Sons, United States.

6. R.I. McKay. 2000. *Fitness Sharing in Genetic Programming, proc.* GECCO-2000, San Francisco.

7. M. Rinaldi. 2005. *Matematika Diskrit*, Informatika ITB, Bandung.

8. M. Rinaldi. 2006. *Strategi Algoritmik,* Informatika ITB, Bandung.

9. R.A. Amin. 2007, *Travelling Salesman Problem*, http://informatika.org/~rinaldi/stmik/Makalah/MakalahStmik30.pdf, diakses pada tanggal 28 Desember 2009.

10. J.B. Richard. 2001. *Discrete Mathematics,* Prentice Hall International, Chicago.

11. S. Bruno. 1998. *Fitness Sharing and Niching Methods Revisited,* IEEE, Transactions on Evolutionary Computation 2 (3) 23-28.

12. S. R. Forrest, A. Perelson. 1993. Searching for diverse, cooperative subpopulations with genetic algorithms, Evolutionary Computation.

13. S. Admi. 2007. Hybrid Genetic Algorithm dengan Fuzzy Logic Controller: Sebuah Pendekatan Baru Penyelesaian Travelling Salesman Problem (TSP), Universitas Lampung, Lampung.