**GAZİ ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**


# BM469-GENETİK ALGORİTMALAR VE PROGRAMLAMA

## PROJE RAPORU

**Fatma Nur KORKMAZ - 191180058**

**Ocak 2024**

# ÖZET

Bu raporda  Fei Liu ve Guangzhou Zeng tarafından yazılmış “Study of genetic algorithm with reinforcement learning to solve the TSP” [1] makalesinin incelemesi sunulmuştur. Bu makale genel olarak ele alınan Gezgin Satıcı Problemi (TSP) nedir ve bu probleme getirilmiş çözüm önerileri üzerine durmuş ve bu problem için yeni bir çözüm önerisi sunulmuştur. Bu çözüm önerisi daha sonra denenmiş ve diğer çözüm önerileri ile karşılaştırmalar yapılarak bu deneylere yönelik sonuçlar sunulmuştur.

# 1. GİRİŞ

Verilen bir şehir kümesi ve aralarındaki mesafeler göz önüne alındığında, Gezinici Satıcı Problemi (TSP), her şehri tam olarak bir kez ziyaret ederek tamamlanan, minimum maliyetli bir yol bulmaktır. TSP, imalat planlama ve VLSI yönlendirme gibi birçok gerçek dünya uygulamasına sahip tanınmış bir NP-hard problemdir. Problemi çözmek için birçok yöntem geliştirilmiştir; ancak kesin algoritmaların karmaşıklığı genellikle üstel düzeydedir. Daha büyük TSP örnekleriyle etkili bir şekilde başa çıkmak ve hesaplama maliyetini azaltmak için optimal çözümleri bulma amacı gütmeyen, ancak kabul edilebilir bir çalışma süresinde yarı-optimal çözümleri bulma hedefi olan yaklaşık algoritmalar geliştirmek gerekmektedir. Makalede, bir şehirden diğerine olan mesafenin, karşıt yöndeki mesafeyle aynı olduğu simetrik TSP ele alınmıştır. Bu sebeple bu raporda da simetrik TSP ele alınacaktır.

Makalede, genellikle yaklaşık algoritmaların iki kategoriye ayrılabileceği belirtilmiştir: yerel arama ve global arama yaklaşımları. Yerel arama yaklaşımları, 2-opt, 3-opt ve Lin-Kernigan gibi yöntemleri içerir ve genellikle geometrik komşuluk bilgisine dayandıkları için etkili ve hızlı bir yakınsamaya sahiptir. Ancak, çözümlerin çeşitliliğiyle ilgilenmedikleri için yerel minimumlarda takılı kalabilirler. Genetik Algoritma, TSP gibi büyük arama alanlarına sahip problemler için uygun olan bir global arama algoritmasıdır. Ancak, genetik operatörler problem spesifik olduğundan, iyi performans için problem spesifik çaprazlama operatörleri tasarlanmalıdır.Makale, EAX (edge assembly crossover) adlı bir çaprazlama yöntemini incelemekte ve H-EAX adlı problem spesifik bir çaprazlama yöntemi önermektedir. Ardından, taksonomi ve çeşitli yaklaşımların performanslarını karşılaştıran bir literatür taraması yapmaktadır. Reinforcement Learning (RL), TSP çözümüne yönelik olarak kullanılmış ve RL operatörünün problem bağımsız olduğunu belirtmiştir. RL'nin global arama yeteneğini artırmak için, koloni öğrenmesi gibi yöntemler önerilmiştir, ancak bu yöntemlerin hesaplama maliyeti genellikle yüksektir.

Son olarak, makalede, RMGA (reinforcement mutate genetic algorithm) adlı bir algoritmanın tanıtımı yapılmaktadır. Bu algoritma, EAX ve RL'yi birleştirerek TSP çözümü için önerilmiştir. RMGA'nın tasarımında, EAX'dan elde edilen bireylerin Q-öğrenme kullanılarak ayarlanması, "reinforcement mutation" olarak adlandırılmaktadır. Bu algoritmanın, EAX-GA ve LK-Helsgaun yöntemleri ile karşılaştırıldığı deneysel sonuçlar, RMGA'nın etkili bir strateji ve performansa sahip olduğunu göstermektedir.

## 2. DİĞER YÖNTEMLER

Gezgin satıcı problemi için geliştirilen birçok çözüm vardır. Makalede bahsedilen çözümler EAX yöntemi ve HeSEA yöntemleridir.

***EAX Yöntemi:*** TSP problemlerini çözmek için iyi bir çaprazlama operatörü kenar tabanlı olmalı ve 'iyi kenarları' ekleyip koruma mekanizmalarına sahip olmalıdır. EAX, bu gereksinimleri karşılayan iyi bir çaprazlama operatörü olarak kabul edilir.EAX, iki önemli özelliğe sahiptir - ebeveynlerin 'iyi kenarlarını' yeni bir teknikle koruma ve yeni kenarları açgözlü bir yöntemle eklemek.

EAX-GA'nın temel işlemi şu şekildedir:

I. Rastgele seçilen iki ebeveyn birey A ve B'yi seçin, alt-graf $G_{AB}$ yi oluşturun.

II. Adımları (1)-(5) $N_{cross}$ kez tekrarlayın:

(1) $G_{AB}$ içinde A ve B'den sırasıyla kenarları seçerek $AB_{cycle}$ oluşturun. C'nin oluşturulmuş $AB_{cycle}$ olduğunu varsayalım.

(2) $AB_{cycle}$'ı rastgele seçin. D'nin seçilen ABcycle'ların birleşimi olduğunu varsayalım.

(3) $E_A \rightarrow (E_A \cup D) \cup (E_B \cap D)$ ile ara çözümü oluşturun.

(4) Ara çözümü açgözlü stratejiyle geçerli bir tura dönüştürün.

(5) Optimum turu tahmin edin ve hatırlayın.

III. A, B ve turdan en iyi iki bireyi seçip bir sonraki nesil olarak belirleyin.

***HeSEA Yöntemi:*** HeSEA, EAX genetik algoritmasının orijinal seçim mekanizması olan heterojen eşleştirme seçimi (HpS) yerine rastgele eşleştirme seçimi (RpS) kullanmıştır. Aile rekabeti ve heterojen eşleştirme seçimi yoluyla kenar birleştirme çaprazlaması (EAX) ve Lin-Kernighan (LK) yerel aramasını entegre etmiştir. Bu algoritma, özellikle büyük TSP'yi çözmede evrimsel algoritmalar için faydalıdır, ancak LK algoritması ve L kez mutasyon

prosedürü nispeten karmaşıktır. Ayrıca aile rekabeti uzunluğu L'nin teorik bir temeli yoktur.HeSEA algoritması, EAX algoritmasındaki eşleştirme seçim mekanizmasını değiştirmiş ve yerel arama algoritması ile entegre etmiştir. HeSEA algoritması faydalı olsa da, karmaşıktır ve teorik temelleri eksiktir.

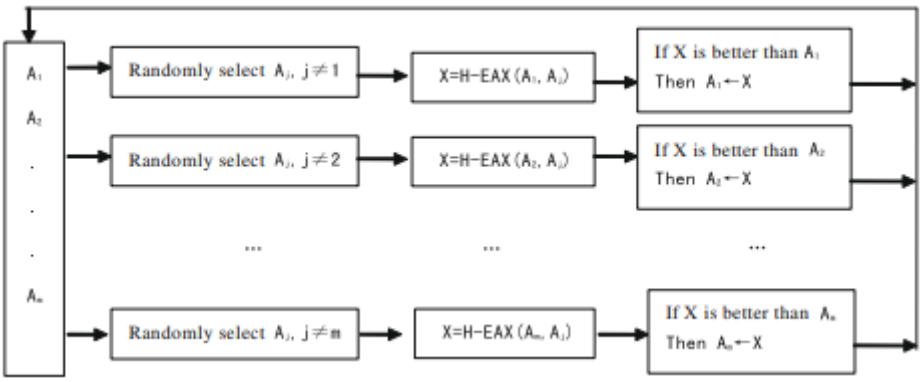
## 3. YÖNTEM

Makalede TSP için yeni bir çözüm önerisi sunulmuştur. Bu çözüm önerisinde çaprazlama için EAX geliştirilerek H-EAX ve mutasyon için RL-M kullanılmıştır. Sonuç olarak bu algoritmalar kullanılarak RMGA algoritması geliştirilmiştir.

***H-EAX:*** Heterogeneous Edge Crossover (Heterojen Kenar Çaprazlaması) anlamına gelir.EAX çaprazlama operatörünün geliştirilmiş bir versiyonudur. EAX, ebeveyn bireylerin kenarlarını korurken yeni kenarlar ekleyerek çeşitliliği artırmayı amaçlar. H-EAX ise, EAX'ın yanı sıra, ebeveyn bireylerin kenarlarını daha iyi korumayı amaçlayan bir mekanizma kullanır. Bu mekanizma, ebeveyn bireylerin kenarlarının benzerliği temelinde, onları daha sıklıkla korur. Bu operatör, GA populasyonunun çeşitliliğini hızlı bir şekilde kullanma avantajına sahiptir, bu da yöntemin çok hızlı bir şekilde yakınsamasına neden olur. Küçük ve orta ölçekli birkaç bin şehre kadar olan TSP sorunları için çok iyi çalışır. Ancak daha büyük ölçekli TSP'lerde genellikle GA'nın yerel minimuma sıkışmasına neden olur.

***Algoritma 1: H-EAX(A,m)***

```
Begin
  For (i = 1;i < m; i++) do
    {select A_i ∈ A as father individual, pick up A_j from A − A_i as
    mother individual at random;
      X ← iterate N_cross times EAX(A_i, A_j) and generate the
      best individual;
      If X is better than A_i, then A_i ← X;   /* X substitutes A_i
      and retain A_i in A* */
    }
  Return A;
End.
```

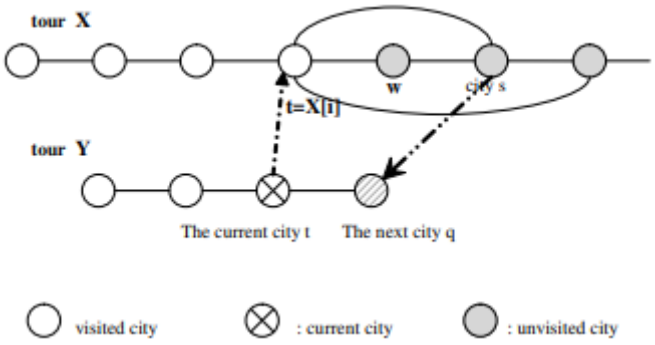



**Şekil 1: H-EAX Algoritma Gösterimi**

***RL-M:*** Reinforcement Learning Mutation (Takviyeli Öğrenme Mutasyonu) anlamına gelir.Takviyeli öğrenme kullanarak bireylerin mutasyonunu gerçekleştirir. Bireylerin yakın komşuluklarını inceleyerek daha iyi çözümler bulmayı hedefler. Bireylerin yakın komşuluklarını inceleyerek daha iyi çözümler bulmayı hedefler. R-LM, bireyleri daha iyi bir yöne doğru yönlendirerek, çözüm kalitesini artırmayı amaçlar.

***Algoritma 2: RL-M(X)***

Begin
Compute the Len $X$ according to DIST;
$\forall(p,q) \in$ Edge, $RQ(p,q) \leftarrow 1/d(p,q)$; /* *initialize the RQ value of Edge* */
Count $\leftarrow 0$;
**Repeat**/* *repeat mutation operation* */
{$t \leftarrow Y[0] \leftarrow X[0]$; /* *initialize the mutation tour* $Y$, $Y[0]$ is the start city, $t$ is the current city in $Y$ */
$J(t) \leftarrow \{1,\ldots,n\} - t$; /* $J(t)$ is the unvisited city sets when the current city is $t$ */
Len $Y \leftarrow 0$;
For ($k = 1$; $k < n$; $k$++)/* *once mutation process* */
{If ($k \neq n-1$) then
{$q \leftarrow$ select city $s \in J(t)$ according to the maximum of $\frac{e^{RQ(t,s)/T}}{\sum_{z \in J(t)} e^{RQ(t,z)/T}}$;
Let the reflected position of the current $t$ in $Y$ to $X$ is $X[i]$ (as showed in Fig. 2)
If $X[i+1] \in J(t)$ and $d(X[i], X[i+1]) < d(t,s)$, $q \leftarrow X[i+1]$;
$Y[k] \leftarrow q$; Len $Y \leftarrow$ Len $Y + d(t,q)$;
If ($k < n-2$) then $J(q) \leftarrow J(t) - q$;
If ($k = n-2$) then $J(q) \leftarrow J(t) - q + Y[0]$;
}
Else
{$q \leftarrow Y[0]$; Len $Y \leftarrow$ Len $Y + d(t,q)$;}
EndIf
$RQ(t,q) \leftarrow (1-\alpha)RQ(t,q) + \alpha[1/d(t,q) + \gamma \max_{z \in J(q)} RQ(q,z)]$; /* update RQ table */
$t \leftarrow q$;
} /* *complete once mutation process* */
Count++;
} **Until** (**Len Y < Len X or Count = N**); /* *N denotes the mutation times* */
If Len $Y <$ Len $X$ then $X \leftarrow Y$, otherwise retain $X$; return($X$, Len $X$);
End.




**Şekil 2:RL-M Algoritma Gösterimi**

***RMGA:*** H-EAX ve RL-M operatörlerini birleştirerek çözüm alanını genişletir, kalitesini ve çeşitliliğini artırır, ve yerel optimumlara takılma riskini azaltır. Gezgin Satıcı Problemi (TSP) için geliştirilmiş bir hibrit genetik algoritmadır. İlk olarak, mevcut populasyondaki birey 'aile babası' olur, ve 'aile babası' hariç populasyondan seçilen başka bir birey 'aile annesi' olur. İkinci olarak, bu çaprazlamadan elde edilen soy 'aile babasından' daha iyi ise, bu soy 'aile babasının' yerine geçer. Bu değiştirme stratejisi, mevcut populasyonun 'çaprazlama soyu' ve orijinal bireyden oluşan bir karışık popülasyon olmasını sağlar. Ve ardından seçilen bir sonraki 'aile annesi', bir sonraki 'aile babası' ile eşleştirildiğinde daha iyi bir birey olabilir. Bu nedenle, karışık popülasyonun avantajı, çaprazlama işleminin genellikle mükemmel bir popülasyonda gerçekleştirilmesini sağlar. Bu, sadece çok benzer ebeveynlerin çaprazlamasını önlemekle kalmaz, aynı zamanda popülasyonun çeşitliliğini de korur. Bu stratejiye göre, iyileştirilmiş EAX algoritması H-EAX, Giriş olarak mevcut populasyon A, Çıkış olarak çaprazlama A ve m'nin popülasyon boyutu olacak şekilde Algoritma 1'de tasvir edilmiştir.

***Algoritma 3: RMGA***

```
Begin
  A_curr ← generate random population of m individuals {A_1,
  A_2,...,A_m} (suitable solutions for the problem);
  Repeat
    {A_next ← Φ;
      C ← H-EAX(A_curr,m);   /*crossover*/
      For (i = 1; i < m; i++)
        {A_i ← RL-M(C_i); /*mutation*/
      A_next ← A_next ∪ A_i;
  }
  A_curr ← A_next;
  }Until the stopping criterion is satisfied;
  Return the optimal tour in A_curr and its length;
  End.
```

## 4. SONUÇLAR VE TARTIŞMALAR

Algoritmanın TSP'yi çözmek için performansı iki açıdan karşılaştırılabilir: çalışma süresi ve sonuç kalitesi. Çalışma süresi, CPU performansı, işletim sistemi ve benzeri gibi çevresel koşullarla ilgili olduğundan aynı çalışma ortamını elde etmek zordur. Bu nedenle makalede, sadece algoritmaların sonuç kalitesi üzerinden karşılaştırma yapılmıştır. Karşılaştırılan algoritmalar EAX-GA ve LKH'dir. TSP örnekleri TSPLIB'den elde edilmiştir.

TSPLIB'deki küçük ve büyük boyutlu TSP örnekleri üzerinde yapılan deneysel sonuçlar, RMGA'nın makul bir süre içinde neredeyse her seferinde en iyi yolu bulabileceğini ve böylece bilinen EAX-GA ve LKH'yi çözüm kalitesi açısından geride bıraktığını göstermiştir.RMGA'nın hibrid yapısı, hem çeşitliliği koruyarak hem de çözüm kalitesini artırarak TSP problemlerini etkili bir şekilde çözebildiğini gösteriyor.

***Tablo 1: Küçük boyutlu TSP'lerde RMGA ve EAX-GA'nın karşılaştırılması***

| TSP instance | RMGA | | EAX-GA | | Population size |
|---|---|---|---|---|---|
| | Error (%) | Best times/total times | Error (%) | Best times/total times | |
| eil101 | 0 | 50/50 | 0.006 | 48/50 | 100 |
| lin318 | 0 | 50/50 | 0.233 | 12/50 | 100 |
| pcb442 | 0 | 50/50 | 0.020 | 26/50 | 100 |
| rat575 | 0.0018 | 44/50 | 0.031 | 3/50 | 300 |
| u724 | 0 | 50/50 | 0.003 | 44/50 | 300 |
| rat783 | 0 | 50/50 | 0.008 | 31/50 | 300 |

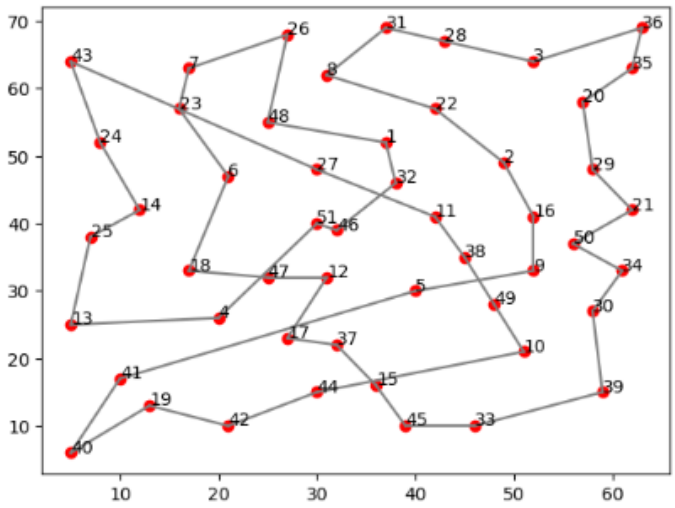
***Tablo 2: N > 1000 olan TSPLIB örnekleri için RMGA ve LKH'nin karşılaştırma sonuçları***

| Name (Optimal) | Method | Best (%) | Average (%) | Worst (%) | $N_{opt}$ | Time (s) | Population size (of RMGA) |
|---|---|---|---|---|---|---|---|
| pr1002 | RMGA | Optimal | Optimal | Optimal | 10/10 | 1442.375 | 100 |
| (259045) | LKH | Optimal | Optimal | Optimal | 10/10 | 4.3 | |
| u1060 | RMGA | Optimal | Optimal | Optimal | 10/10 | 1695.813 | 100 |
| (224094) | LKH | Optimal | Optimal | Optimal | 10/10 | 55 | |
| vm1084 | RMGA | Optimal | Optimal | Optimal | 10/10 | 1305.282 | 100 |
| (239297) | LKH | Optimal | 239318.4(0.0089) | 239407(0.0460) | 7/10 | 24 | |
| pcb1173 | RMGA | Optimal | Optimal | Optimal | 10/10 | 1668.016 | 100 |
| (56892) | LKH | Optimal | 56893(0.0018) | 56897(0.0087) | 8/10 | 15.6 | |
| d1291 | RMGA | Optimal | 50823(0.0433) | 50845(0.0866) | 3/10 | 1084.62 | 100 |
| (50801) | LKH | Optimal | 50826.5(0.0502) | 50886(0.167) | 7/10 | 51.5 | |
| rl1304 | RMGA | Optimal | Optimal | Optimal | 10/10 | 1310.45 | 100 |
| (252948) | LKH | Optimal | 253190.4(0.0958) | 253435(0.1925) | 4/10 | 13.7 | |
| rl1323 | RMGA | Optimal | Optimal | Optimal | 10/10 | 1273.82 | 100 |
| (270199) | LKH | Optimal | 270241.6(0.0158) | 270330(0.0485) | 1/10 | 17.7 | |
| nrw1379 | RMGA | Optimal | 56639.6(0.0028) | 56643(0.0088) | 6/10 | 2574.6 | 300 |
| (56638) | LKH | Optimal | 56638.5(0.00088) | 56643(0.0088) | 9/10 | 28.6 | |
| u1432 | RMGA | Optimal | 152996.1(0.0171) | 153035(0.0425) | 5/10 | 2157.34 | 300 |
| (152970) | LKH | Optimal | Optimal | Optimal | 10/10 | 64.2 | |
| d1655 | RMGA | Optimal | Optimal | Optimal | 10/10 | 3435.23 | 300 |
| (62128) | LKH | Optimal | 62128.1(0.0002) | 62129(0.0016) | 9/10 | 46.3 | |
| vm1748 | RMGA | Optimal | Optimal | Optimal | 10/10 | 1994.38 | 300 |
| (336556) | LKH | Optimal | 336631.5(0.0224) | 336701(0.0431) | 3/10 | 44.9 | |
| u1817 | RMGA | Optimal | 57206(0.0087) | 57222(0.0367) | 4/10 | 2144.69 | 300 |
| (57201) | LKH | Optimal | 57252.3(0.0897) | 57272(0.124) | 1/10 | 85.7 | |
| rl1889 | RMGA | Optimal | 316539.5(0.0011) | 316550(0.0044) | 7/10 | 4584.98 | 300 |
| (316536) | LKH | Optimal | 316539.8(0.0012) | 316549(0.0041) | 5/10 | 57 | |
| d2103 | RMGA | Optimal | 80451.3(0.0016) | 80455(0.0062) | 7/10 | 2919.703 | 300 |
| (80450) | LKH | 80455(0.006) | 80474.7(0.031) | 80490(0.0497) | 0/10 | 404.2 | |
| u2152 | RMGA | Optimal | Optimal | Optimal | 10/10 | 3198.7 | 300 |
| (64253) | LKH | Optimal | 64270.7(0.028) | 64299(0.072) | 4/10 | 147.6 | |
| pr2392 | RMGA | Optimal | Optimal | Optimal | 10/10 | 4243.7 | 300 |
| (378032) | LKH | Optimal | Optimal | Optimal | 10/10 | 125.3 | |

Tablo 1 ve Tablo 2, algoritmaların TSP problemlerini çözme üzerine bir çalışmanın deneysel sonuçlarını özetler. Bu sonuçlar, RMGA'nın EAX-GA ve LKH algoritmalarına kıyasla daha iyi performans gösterdiğini gösteriyor. RMGA, özellikle hata yüzdesi açısından avantaj sağlıyor.deneylere göre, LKH bu boyuttaki iki başka durum için, yani için optimal turları bulamamıştır. Çalışma süresi açısından, RMGA'nın 2392 şehire kadar olan TSP'lerde LKH'den daha yavaş olduğu görülmüştür Bunun nedeni, RMGA'nın güçlendirme mutasyonu için harcadığı zamanın toplam çalışma süresinin büyük bir kısmını oluşturmasıdır. Ancak, RMGA'nın tüm durumlar için elde ettiği sonuç kalitesi, LKH'nin sonuç kalitesinden daha iyidir. RMGA'nın büyük bir kısmı optimal çözüme yaklaşabilir.

Makalede verilen algoritmalara göre python programlama dili ile kodlanmıştır. Bu genetik algoritma, 51_city.txt dosyasında belirtilen 51 şehir arasında gezilecek en kısa yolun bulunması amacıyla kullanılmıştır. Algoritma, önce popülasyonu rastgele oluşturup şehirleri karıştırarak başlamakta ve ardından çaprazlama ve mutasyon operatörleri kullanılarak yeni nesiller üretmektedir. Her nesilde en iyi bireyler elde edilmesi sağlanmıştır. Çalışmanın sonuçlarına göre, 200 nesilde en iyi çözüm, başlangıçta belirlenen 450 birimlik hedefe ulaşarak algoritma erken sonlandırılmıştır. Başlangıçta rastgele popülasyonun iyileştirilmesiyle, çözümün uzunluğu 1347.85 birimden 538.58 birime düşürülmüştür.

Bu durum, genetik algoritmanın TSP problemini çözmek için etkili bir şekilde çalıştığını ve iterasyonlarla çözümün hedefe yaklaştığını göstermektedir. Grafikte çizilen yol, en iyi çözümün şehirleri nasıl ziyaret ettiğini görsel olarak göstermektedir.




**Şekil 3: Programda Bulunan Optimal Yol**

## KAYNAKLAR

[1] Liu F., Zeng G. (2009). “Study of genetic algorithm with reinforcement learning to solve the TSP”. *Expert Systems with Applications*.