Problema comis-voiajorului

O analiză a unei rezolvări cu algoritm genetic

1. Descrierea problemei

Problema comis-voiajorului (TSP – Traveling Salesman Problem în engleză) este o problemă clasică și cea mai frecvent analizată problemă de optimizare combinatorică, cu o gamă largă de metodologii de rezolvare. Avem o listă de orașe și distanțele dintre acele orașe. O persoană are de vizitat acea listă de orașe, începând și sfârșind cu primul oraș din lista dată. Se cere să se găsească turul în care fiecare oraș este vizitat o singură dată și distanța parcursă este minimă.

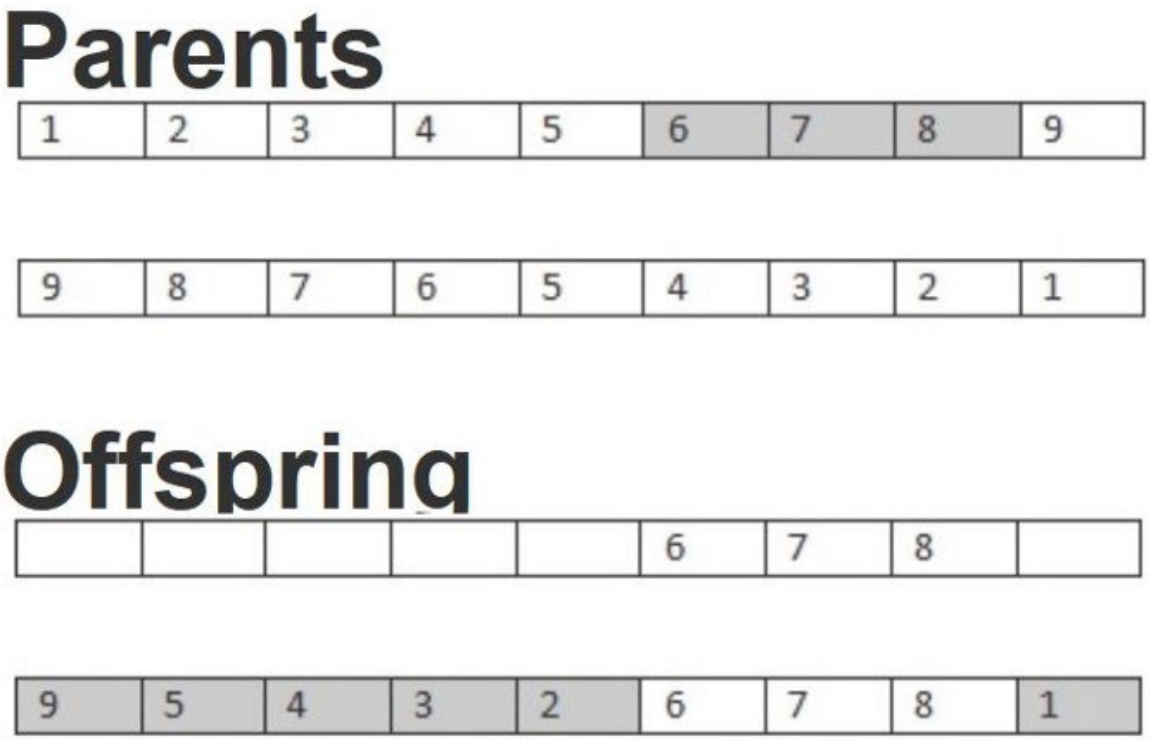
2. Algoritm de rezolvarea – pseudocod

[START] Generăm populația de indivizi (în acest caz, 100 de indivizi per populație, indivizii sunt drumurile care satisfac condițiile menționate în descrierea problemei, iar populația este o colecție de drumuri), această primă populație fiind generația 1.
[FITNESS] Prin intermediul funcției de fitness, evaluăm fiecare individ din populație (pentru TSP, funcția fitness verifică lungimea drumului)
[ELITE] Din populația curentă, păstrăm individul pentru care s-au obținut cele mai bune rezultate în funcția de fitness (deoarece vorbim despre problema comis-voiajorului, suntem interesați să găsim drumul cu lungimea cea mai mică).
[NP] Creăm o populație nouă prin repetarea următoarelor etape, până la obținerea unei noi populații complete; numărul generației va crește cu 1 și noua populație va înlocui vechea populație.

[SELECTION] Selectăm 2 părinți din generația anterioară

[CROSSOVER] Folosind operatorul de încrucișare, încrucișăm cei 2 indivizi aleși ca și părinți ca să obținem un nou individ (selectăm la întâmplare o secțiune a primului părinte ca bază a noului individ și completăm cu elementele celui de al doilea părinte, în afară de elementele din secțiunea preluată de la primul părinte).

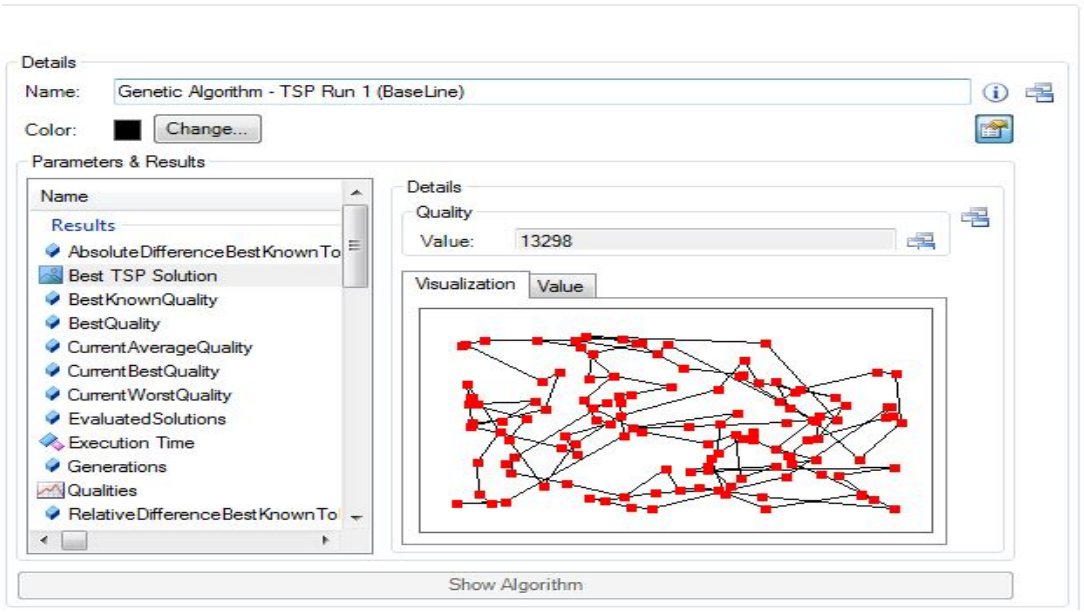
[MUTATION] În funcție de probabilitatea de mutație, există posibilitatea ca noul individ să sufere mutații (două valori din el, luate la întâmplare, vor fi interschimbate). Indiferent dacă s-a produs mutația sau nu, adăugăm noul individ în populație.
[TEST&LOOP] Verificăm numărul de generații; dacă am ajuns la numărul maxim de generații, algoritmul se încheie și returnează cea mai bună soluție din populația curentă; altfel, repetăm pașii de la [FITNESS] până la [NP] inclusiv, cu tot cu sub-pașii lui [NP].



3. Exemple de rulare

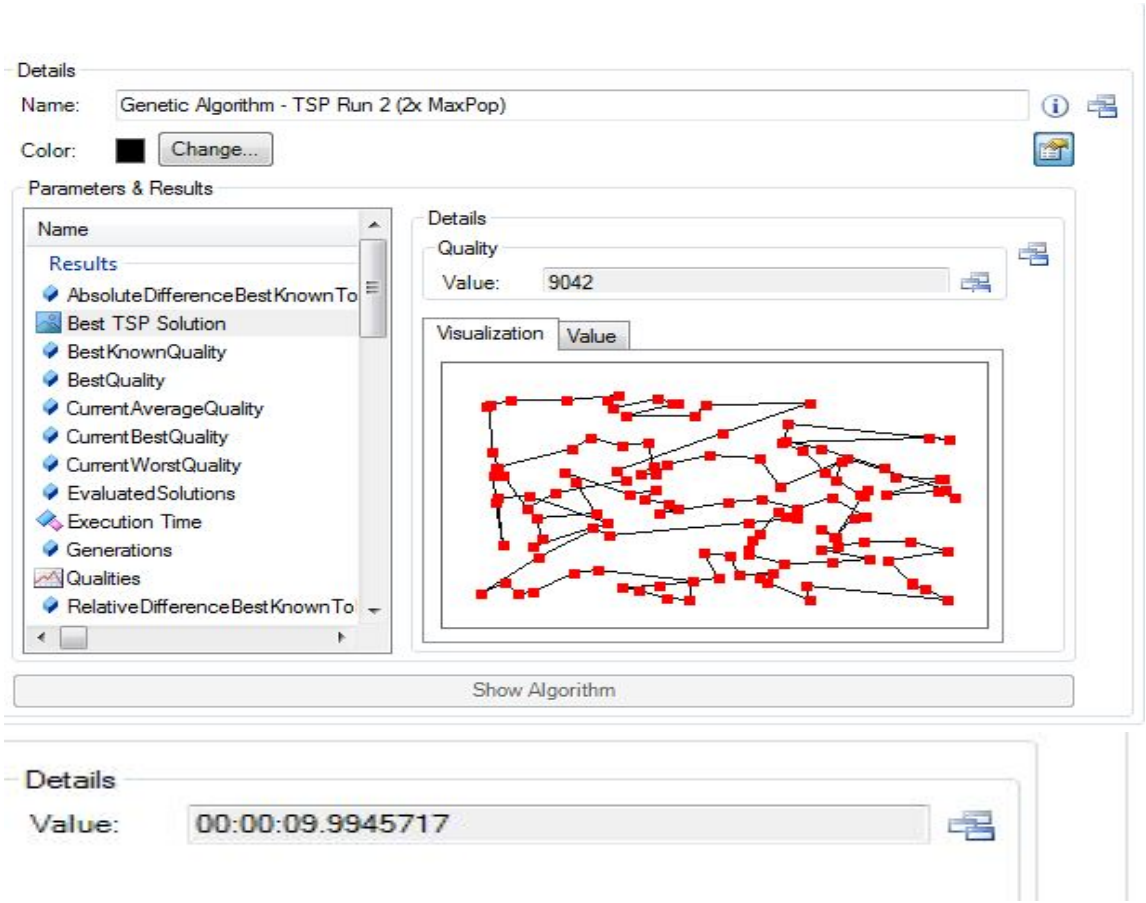
3.1 Rulare cu parametrii de bază

PopulationSize: 100
Elites: 1
MutationProbability: 5%
MaximumGenerations: 1000
Selector: ProportionalSelector
Crossover: OrderCrossover2 (cf. Affenzeller, M. et al. 2009. Genetic Algorithms and Genetic Programming - Modern Concepts and Practical Applications. CRC Press. p. 135)
Mutator: InversionManipulator



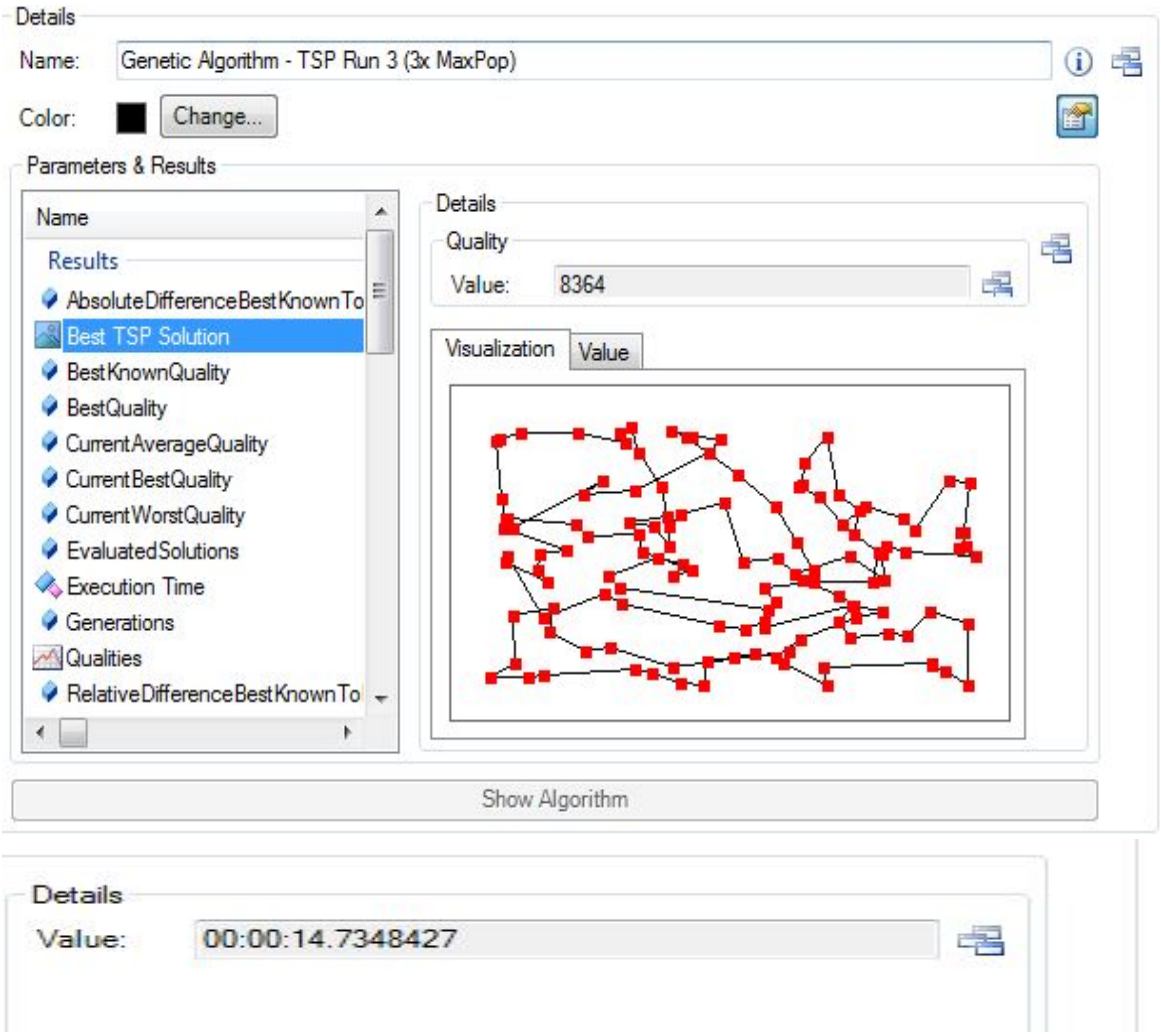
3.2 Rulare cu generație maximă dublă

PopulationSize: 100
Elites: 1
MutationProbability: 5%
MaximumGenerations: 2000
Selector: ProportionalSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



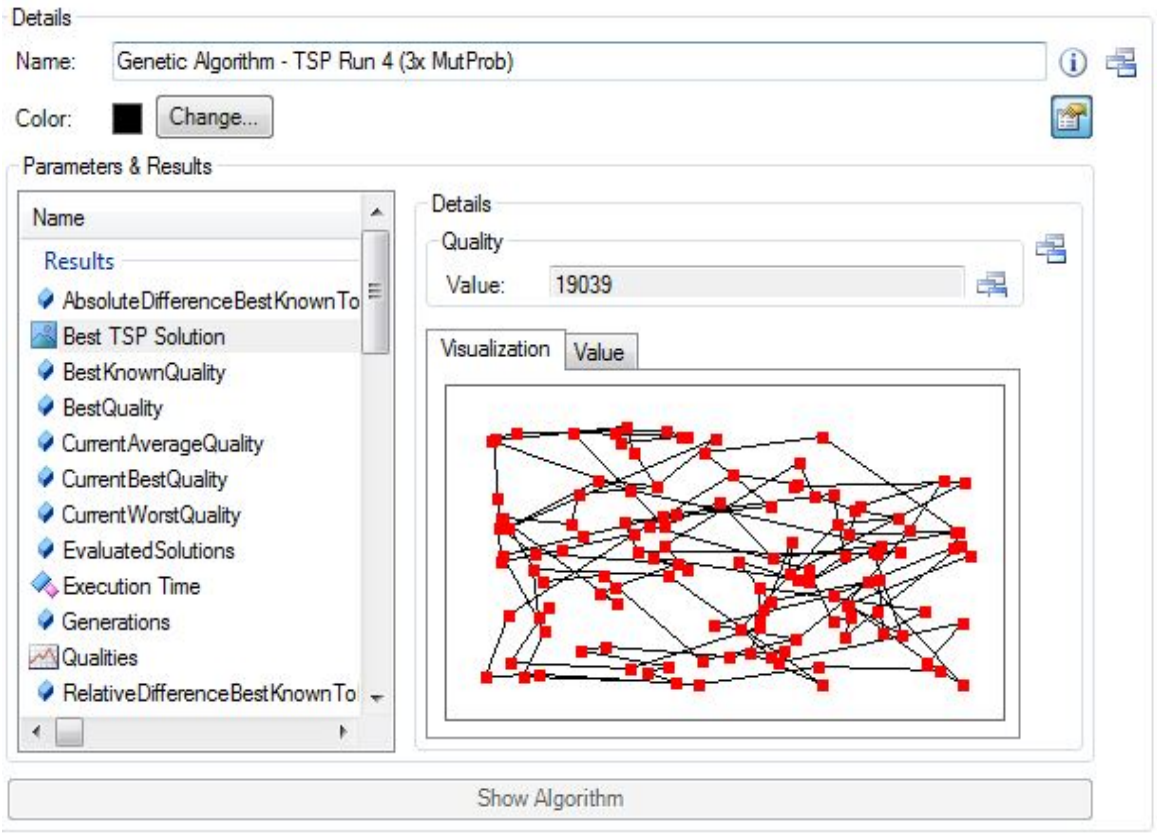
3.3 Rulare cu generație maximă triplă

PopulationSize: 100
Elites: 1
MutationProbability: 5%
MaximumGenerations: 3000
Selector: ProportionalSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



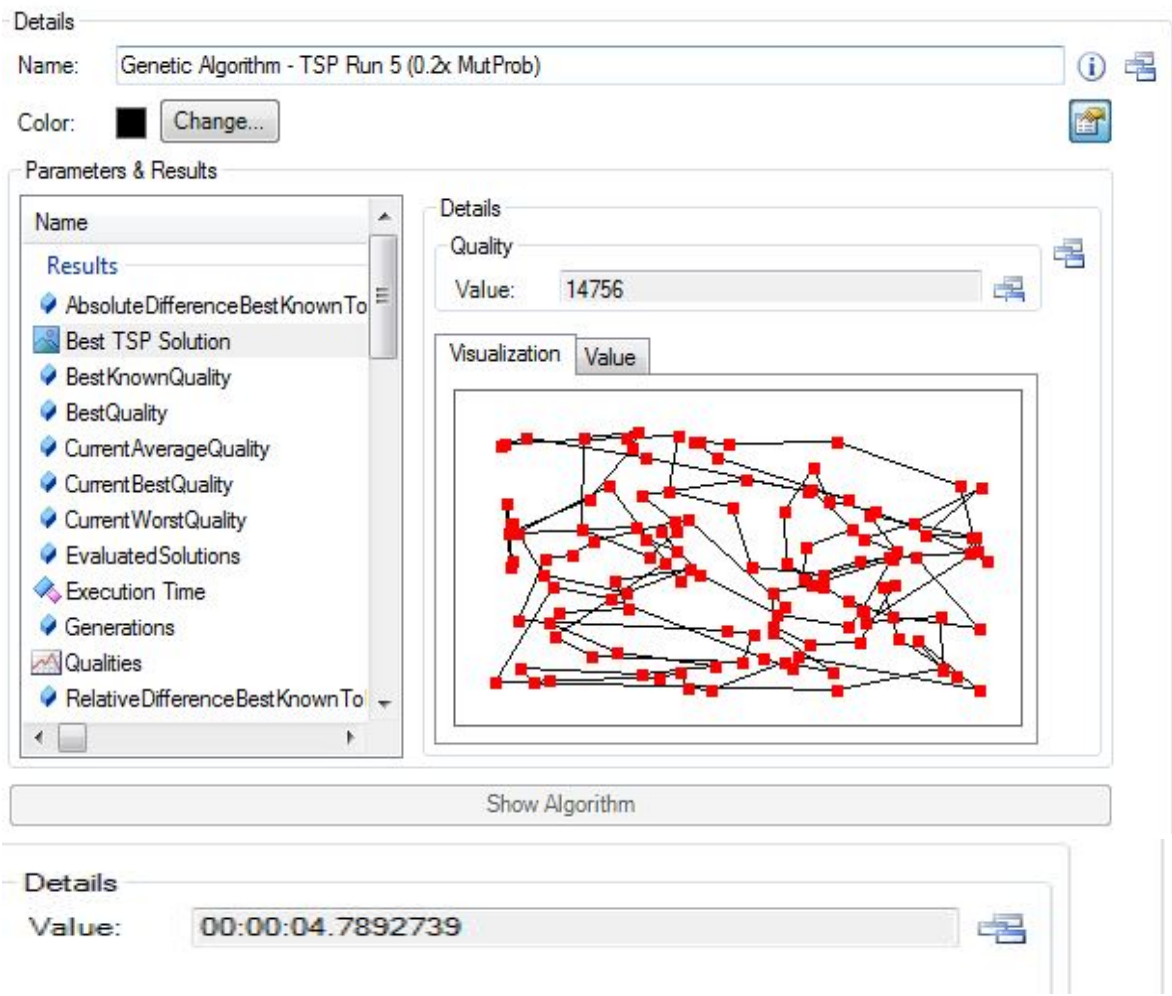
3.4 Rulare cu probabilitate de mutație triplă

PopulationSize: 100
Elites: 1
MutationProbability: 15%
MaximumGenerations: 1000
Selector: ProportionalSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



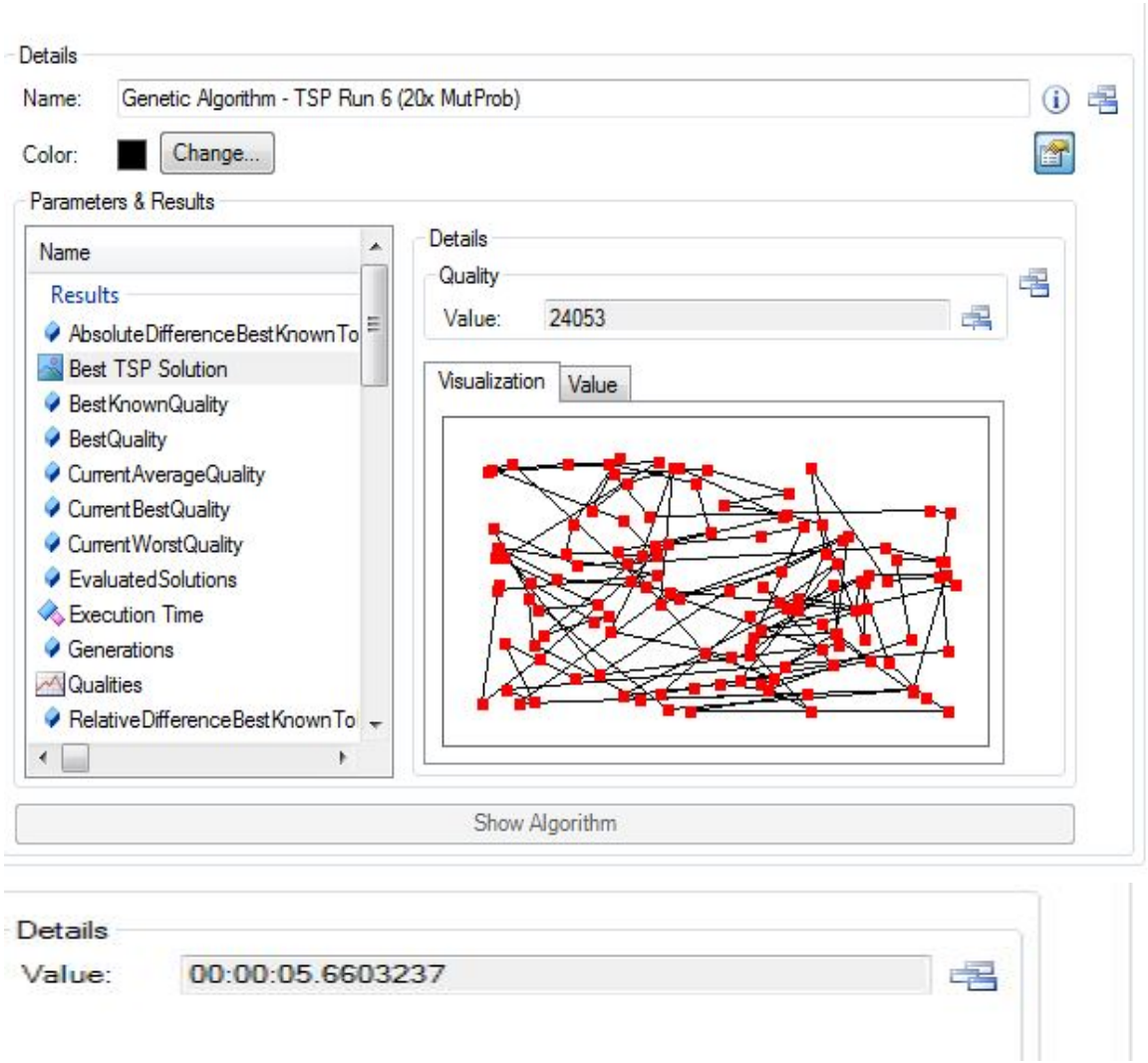
3.5 Rulare cu probabilitate de mutație 1%

PopulationSize: 100
Elites: 1
MutationProbability: 1%
MaximumGenerations: 1000
Selector: ProportionalSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



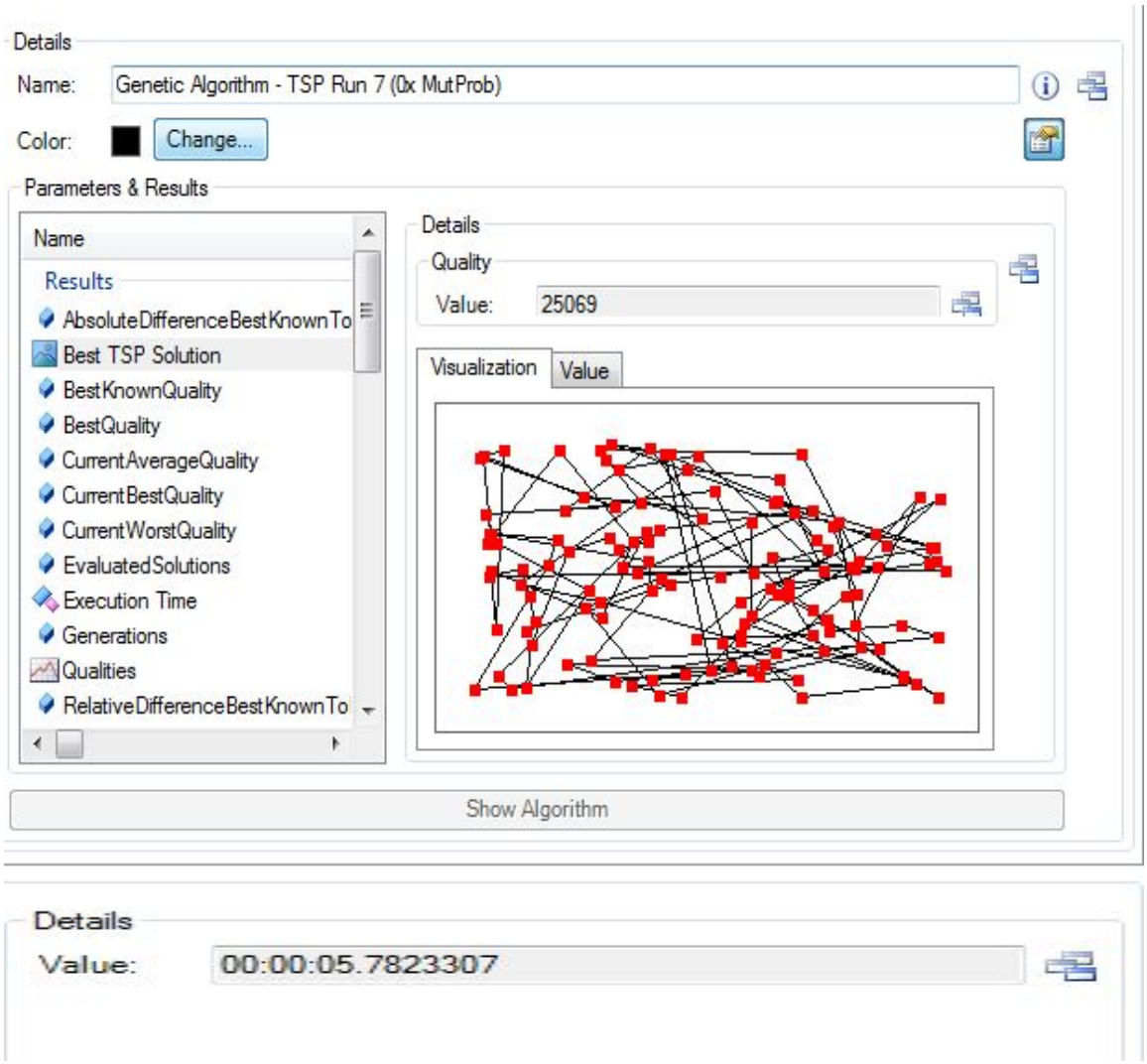
3.6 Rulare cu probabilitate de mutație 100%

PopulationSize: 100
Elites: 1
MutationProbability: 100%
MaximumGenerations: 1000
Selector: ProportionalSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



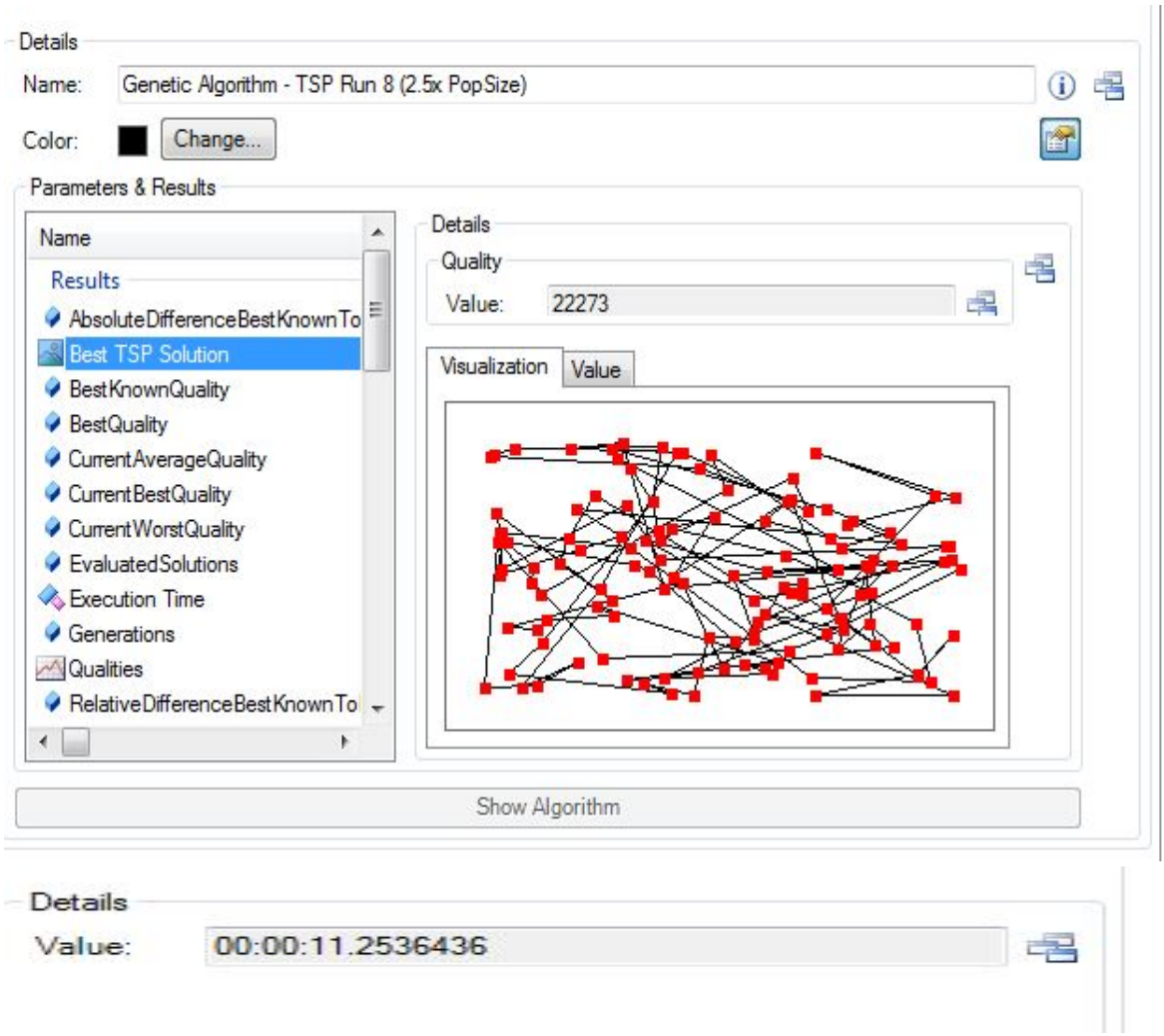
3.7 Rulare cu probabilitate de mutație 0%

PopulationSize: 100
Elites: 1
MutationProbability: 0%
MaximumGenerations: 1000
Selector: ProportionalSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



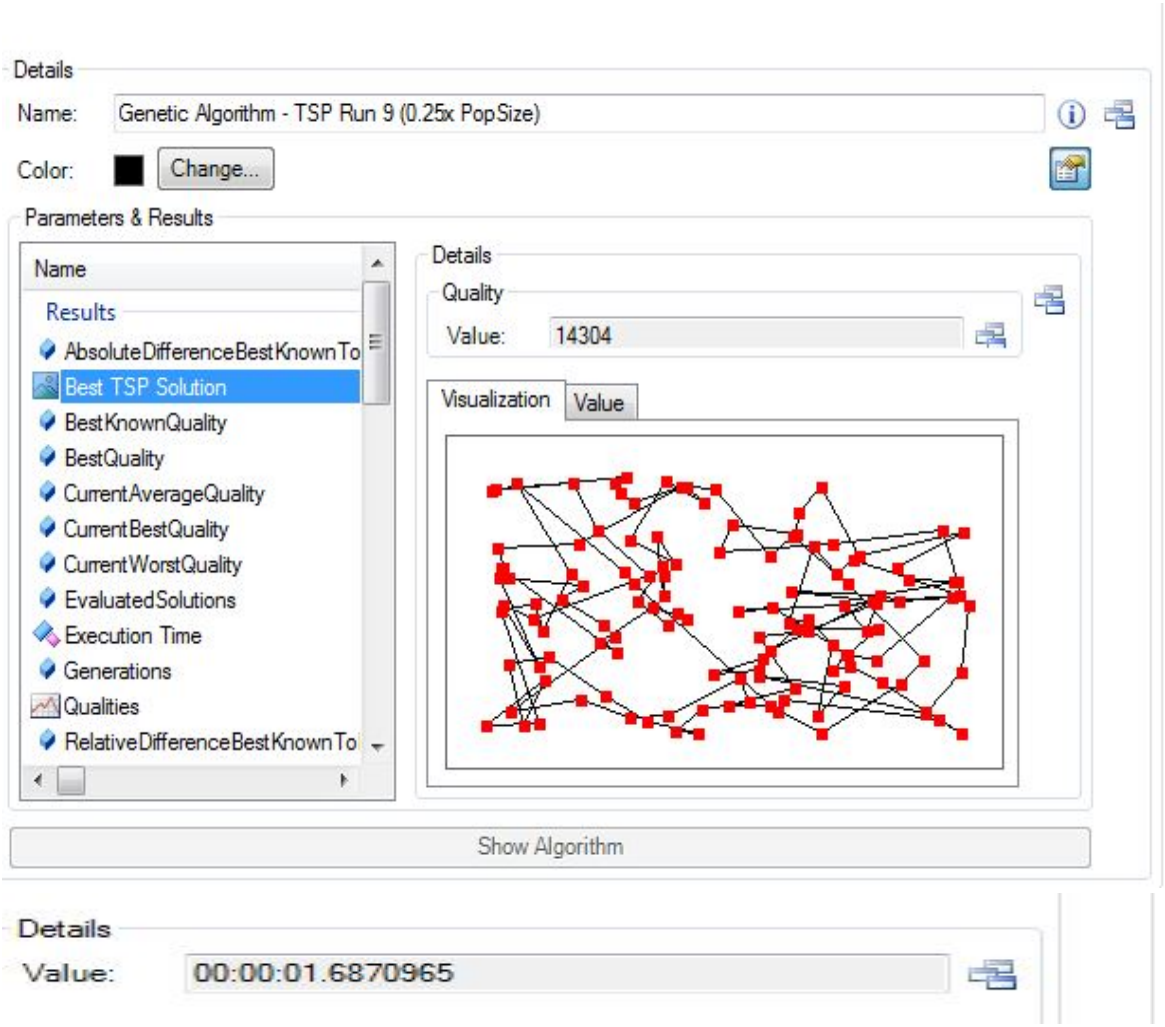
3.8 Rulare cu populație maximă 250

PopulationSize: 250
Elites: 1
MutationProbability: 5%
MaximumGenerations: 1000
Selector: ProportionalSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



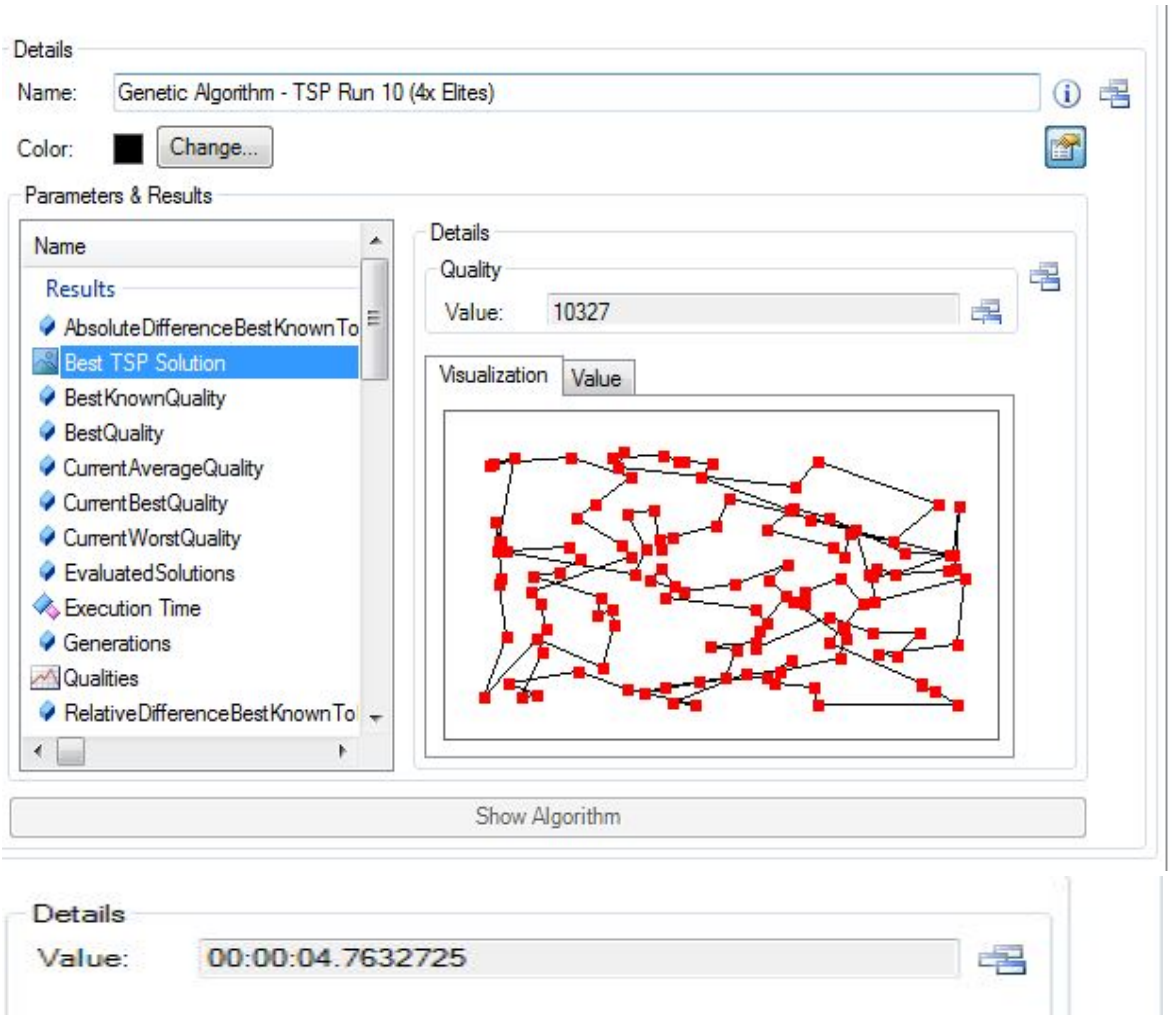
3.9 Rulare cu populație maximă 25

PopulationSize: 25
Elites: 1
MutationProbability: 5%
MaximumGenerations: 1000
Selector: ProportionalSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



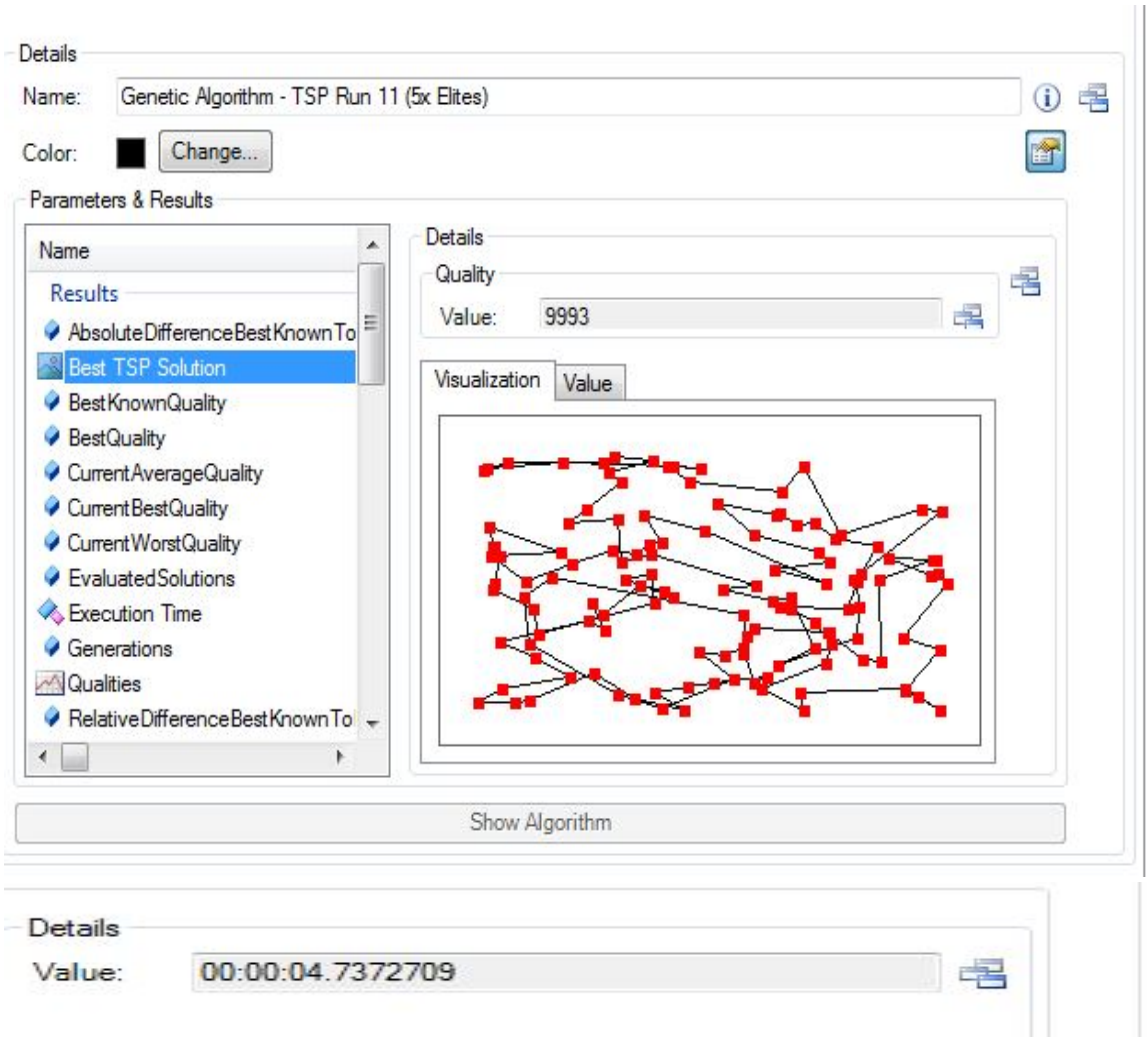
3.10 Rulare cu 4 membrii de elită

PopulationSize: 100
Elites: 4
MutationProbability: 5%
MaximumGenerations: 1000
Selector: ProportionalSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



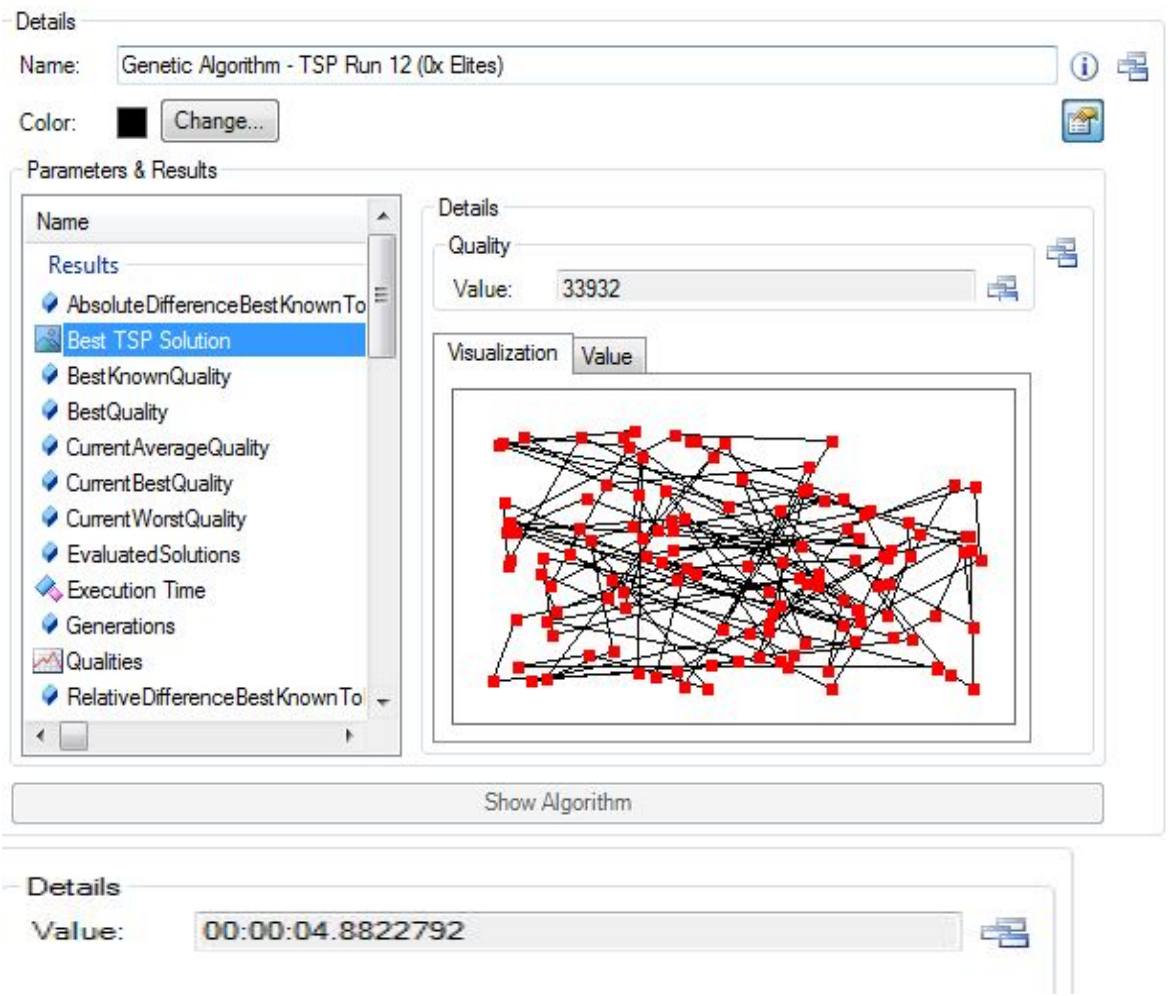
3.11 Rulare cu 5 membrii de elită

PopulationSize: 100
Elites: 5
MutationProbability: 5%
MaximumGenerations: 1000
Selector: ProportionalSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



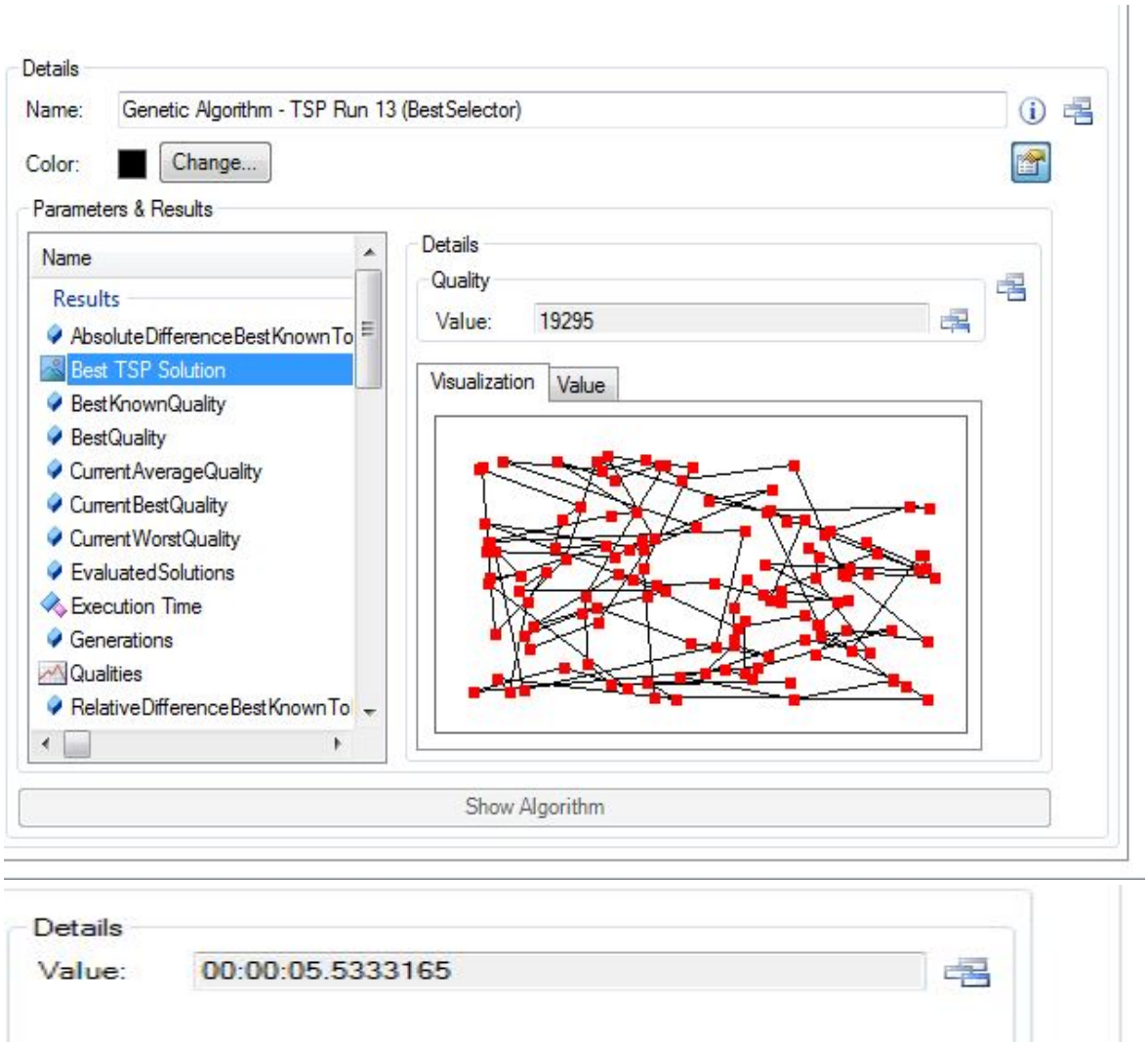
3.12 Rulare fără membrii de elită

PopulationSize: 100
Elites: 0
MutationProbability: 5%
MaximumGenerations: 1000
Selector: ProportionalSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



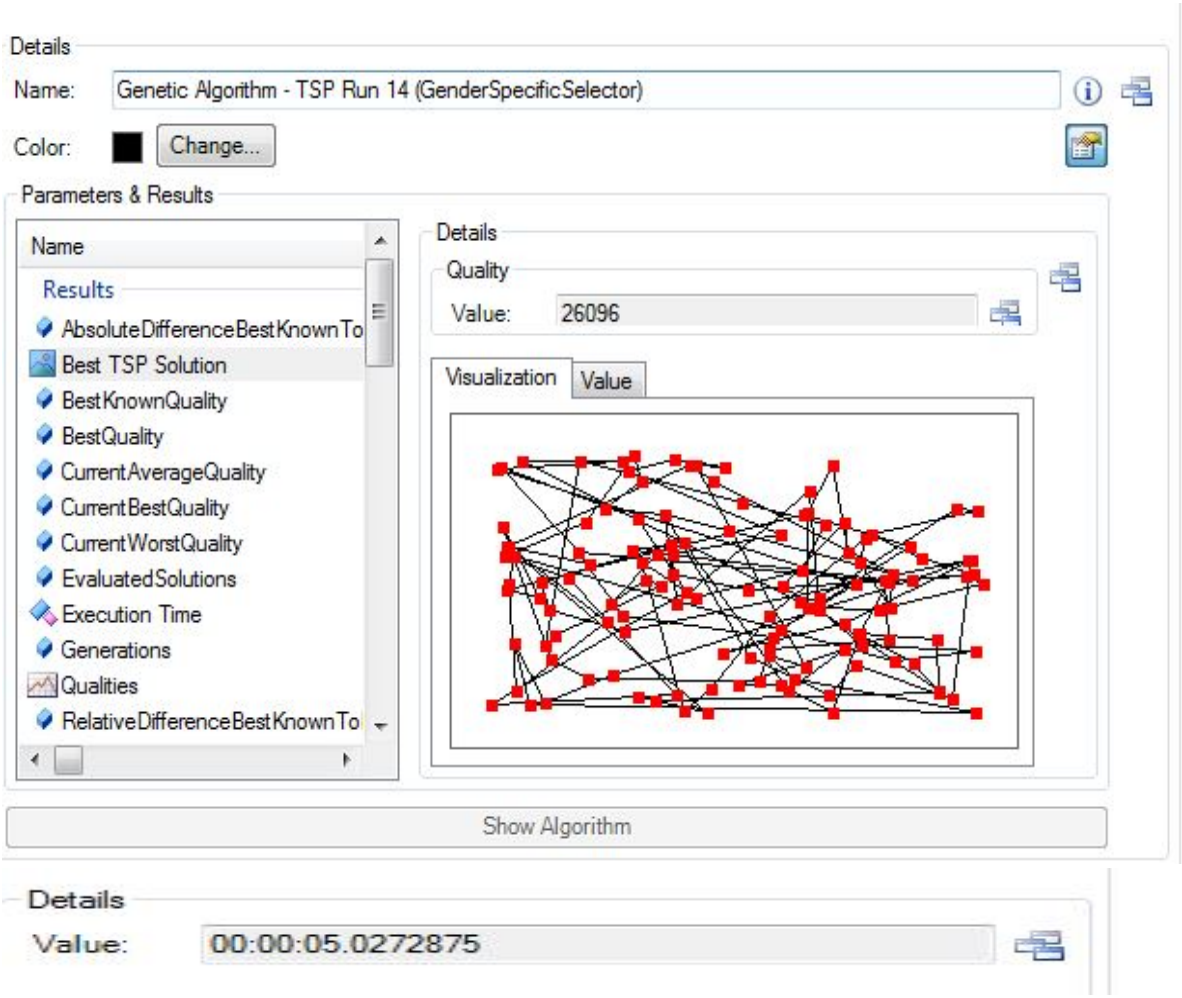
3.13 Rulare cu selector BestSelector

PopulationSize: 100
Elites: 1
MutationProbability: 5%
MaximumGenerations: 1000
Selector: BestSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



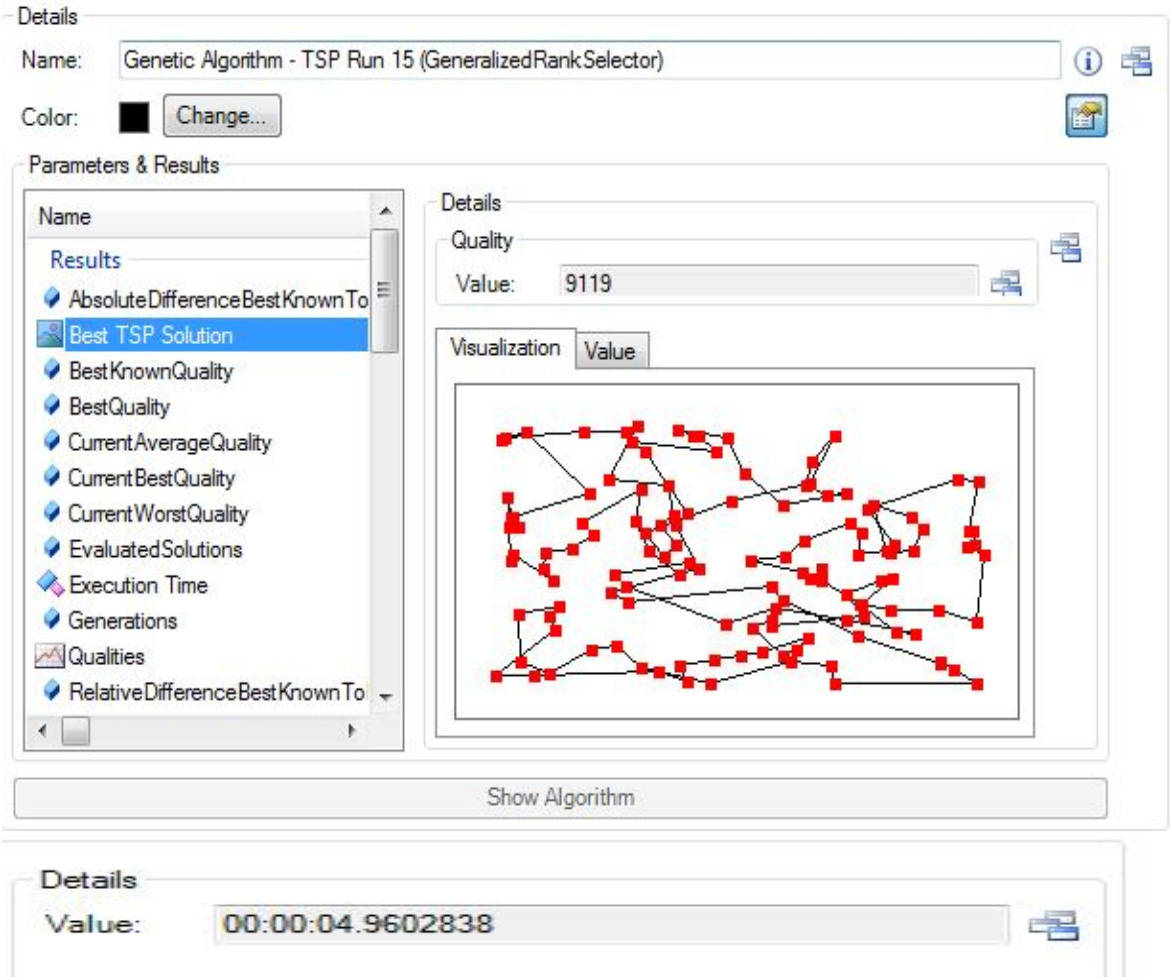
3.14 Rulare cu selector GenderSpecificSelection

PopulationSize: 100
Elites: 1
MutationProbability: 5%
MaximumGenerations: 1000
Selector: GenderSpecificSelection
Crossover: OrderCrossover2
Mutator: InversionManipulator



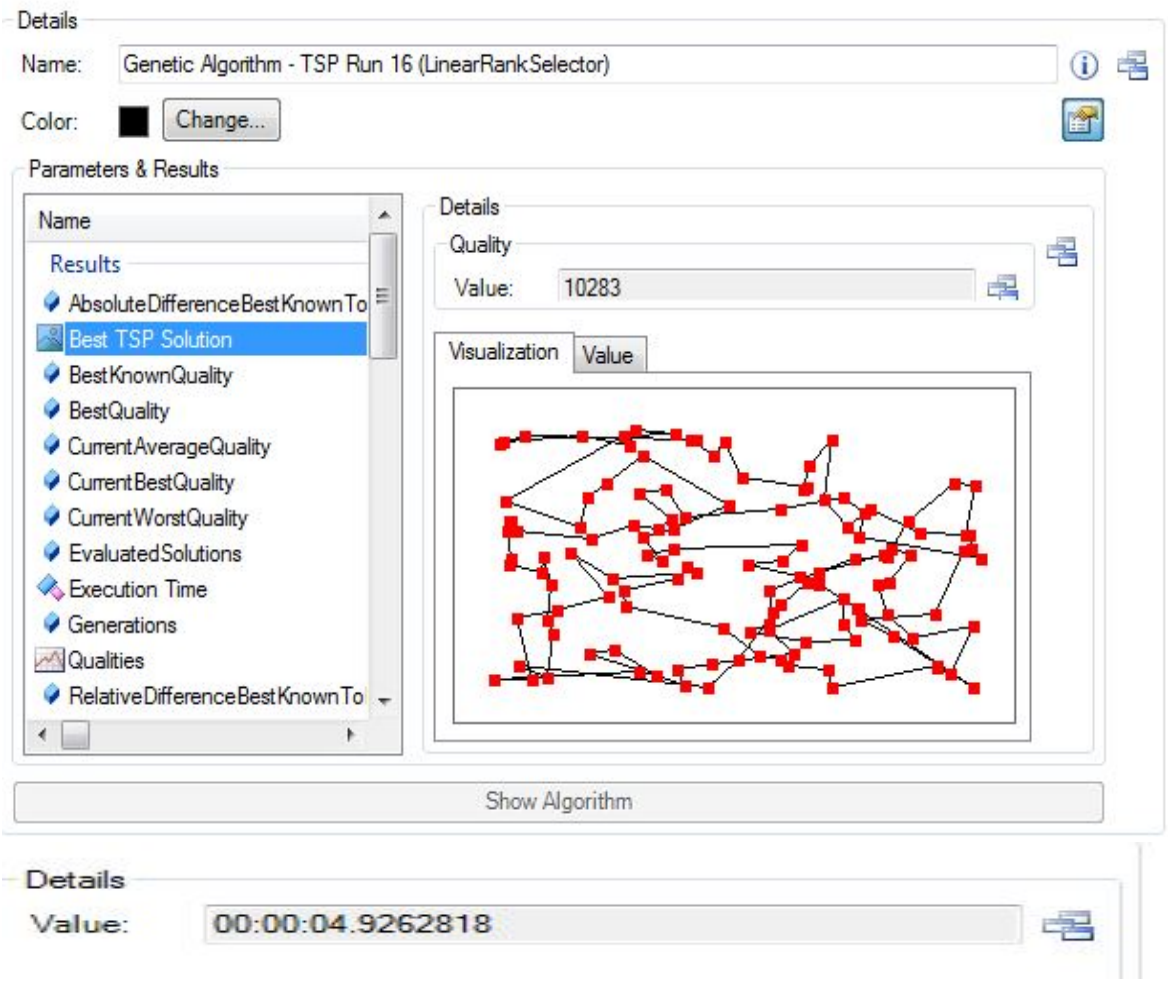
3.15 Rulare cu selector GeneralizedRankSelector

PopulationSize: 100
Elites: 1
MutationProbability: 5%
MaximumGenerations: 1000
Selector: GeneralizedRankSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



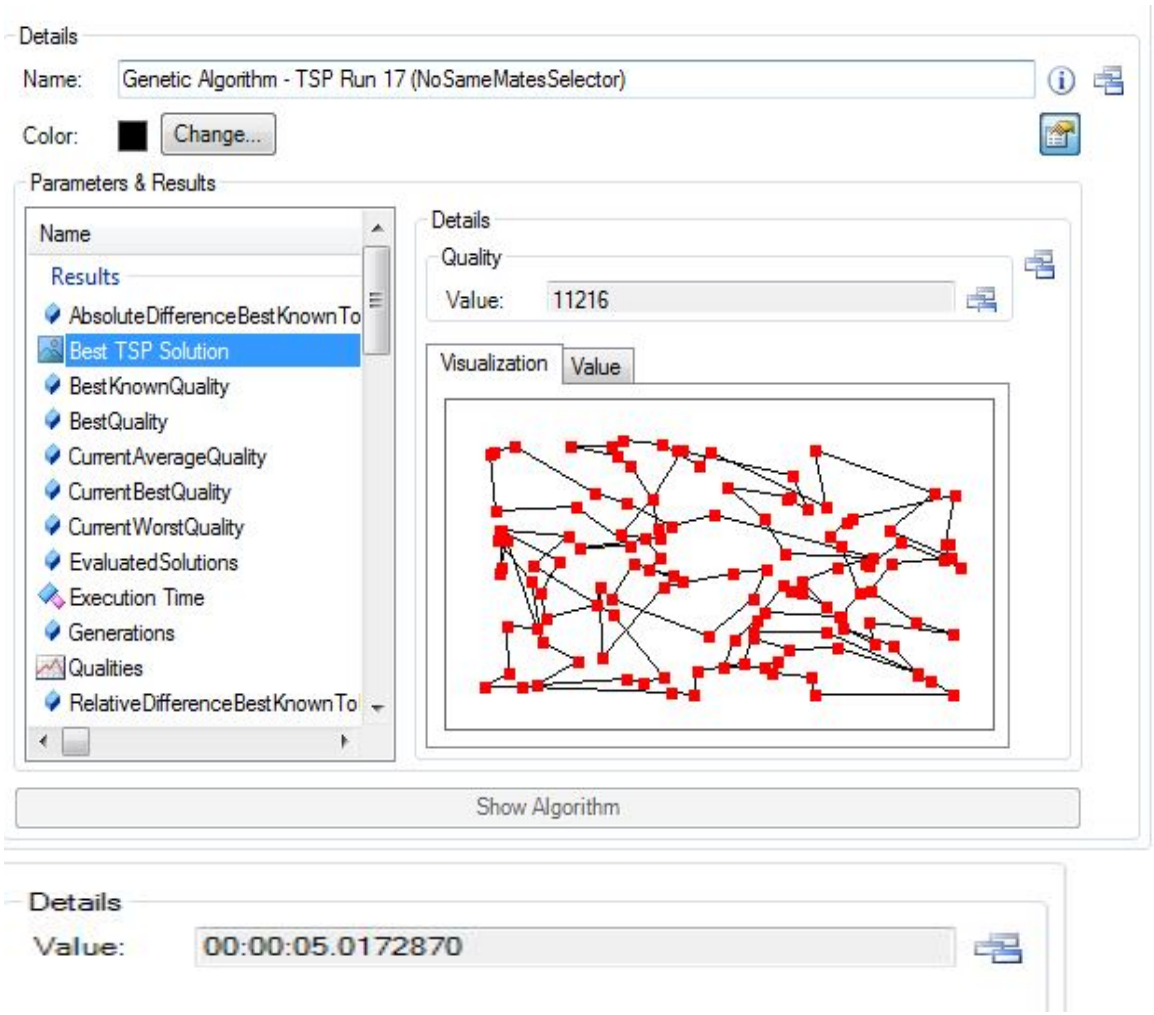
3.16 Rulare cu selector LinearRankSelector

PopulationSize: 100
Elites: 1
MutationProbability: 5%
MaximumGenerations: 1000
Selector: LinearRankSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



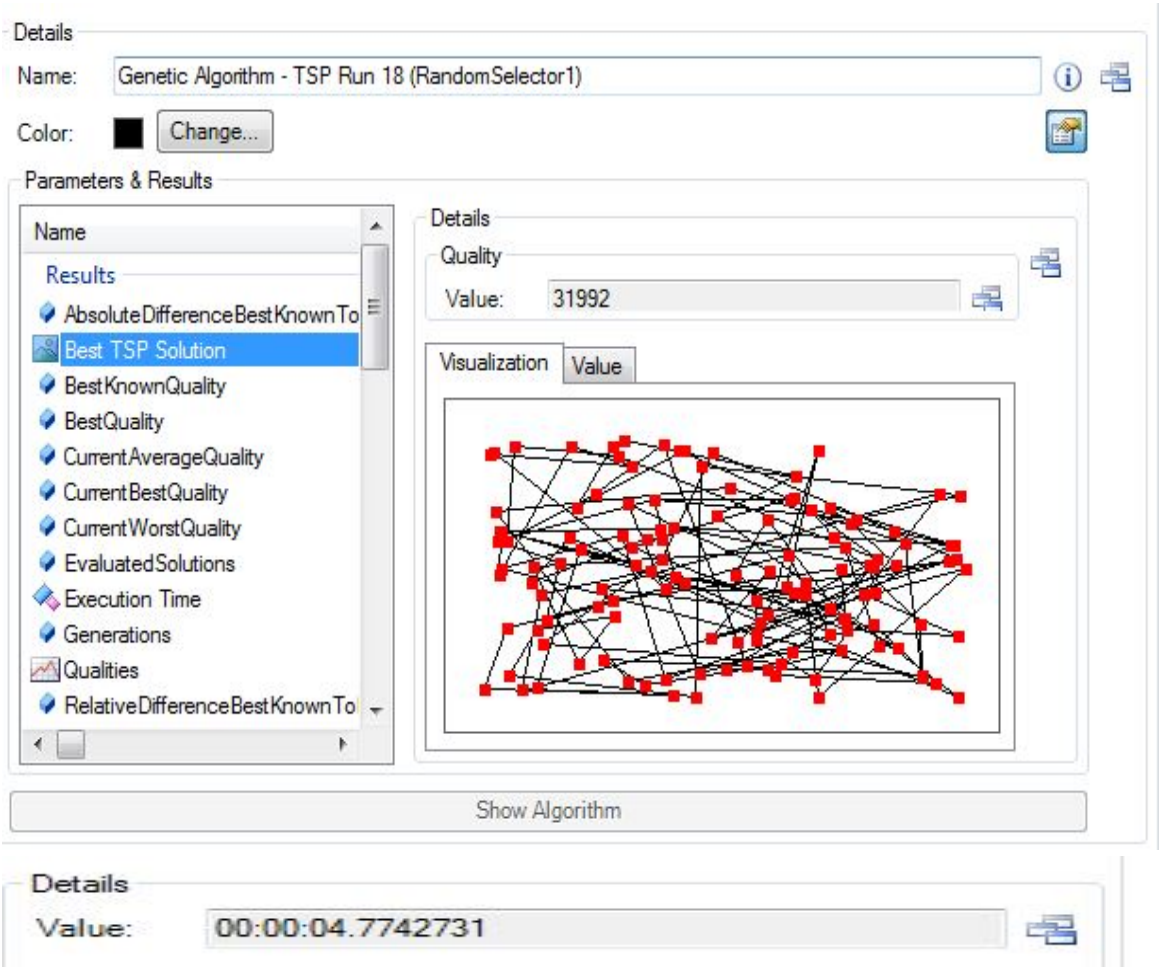
3.17 Rulare cu selector NoSameMatesSelector

PopulationSize: 100
Elites: 1
MutationProbability: 5%
MaximumGenerations: 1000
Selector: NoSameMatesSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



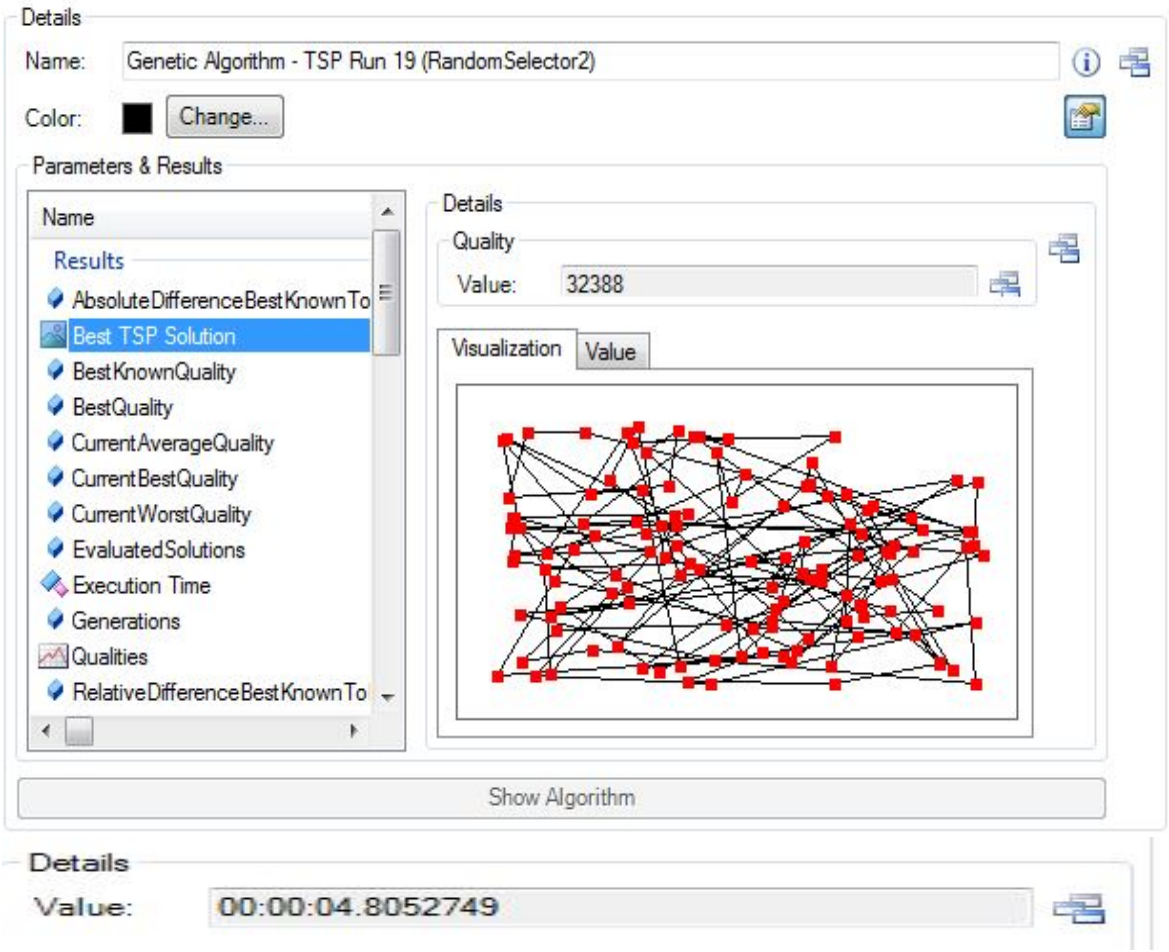
3.18 Rulare cu selector RandomSelector #1

PopulationSize: 100
Elites: 1
MutationProbability: 5%
MaximumGenerations: 1000
Selector: RandomSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



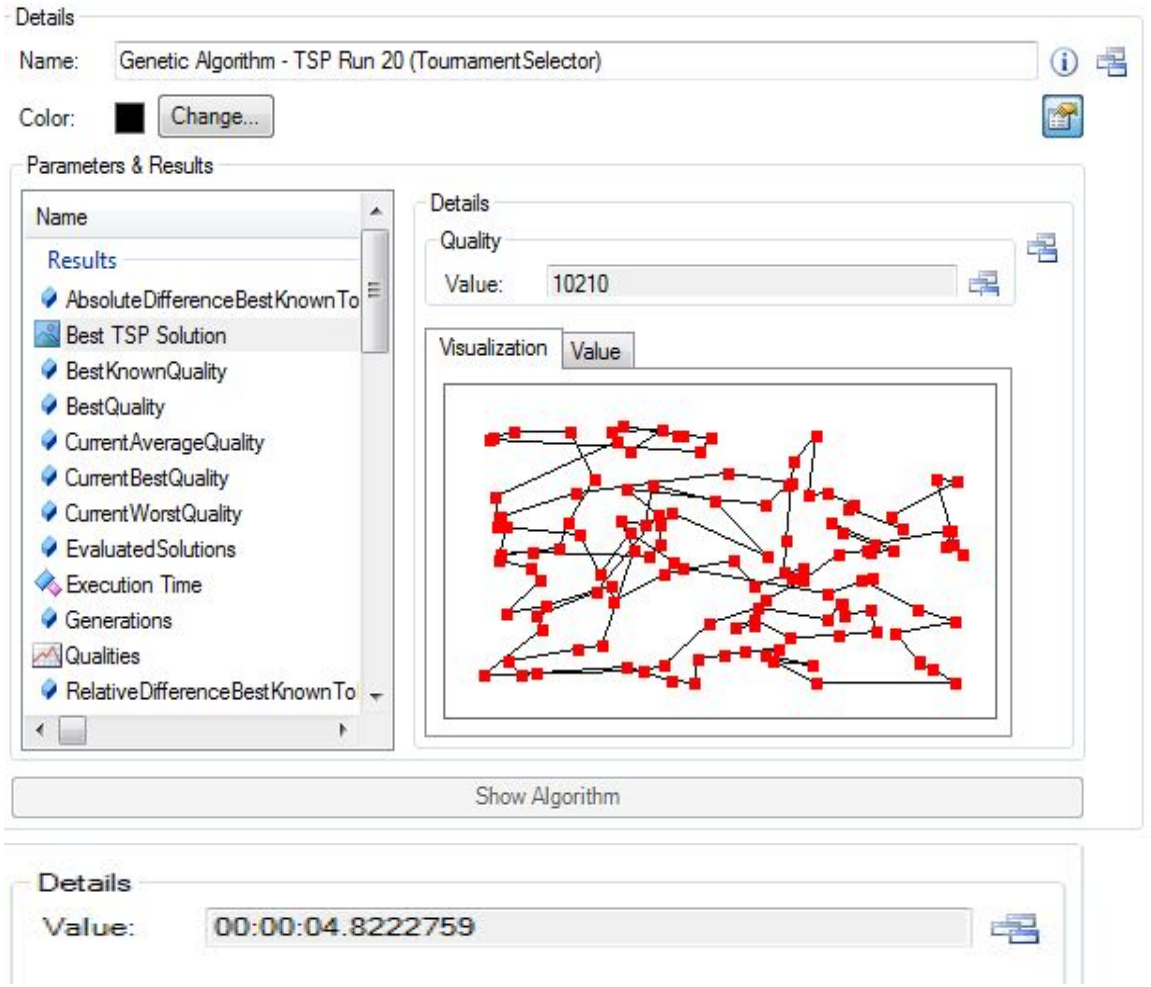
3.19 Rulare cu selector RandomSelector #2

PopulationSize: 100
Elites: 1
MutationProbability: 5%
MaximumGenerations: 1000
Selector: RandomSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



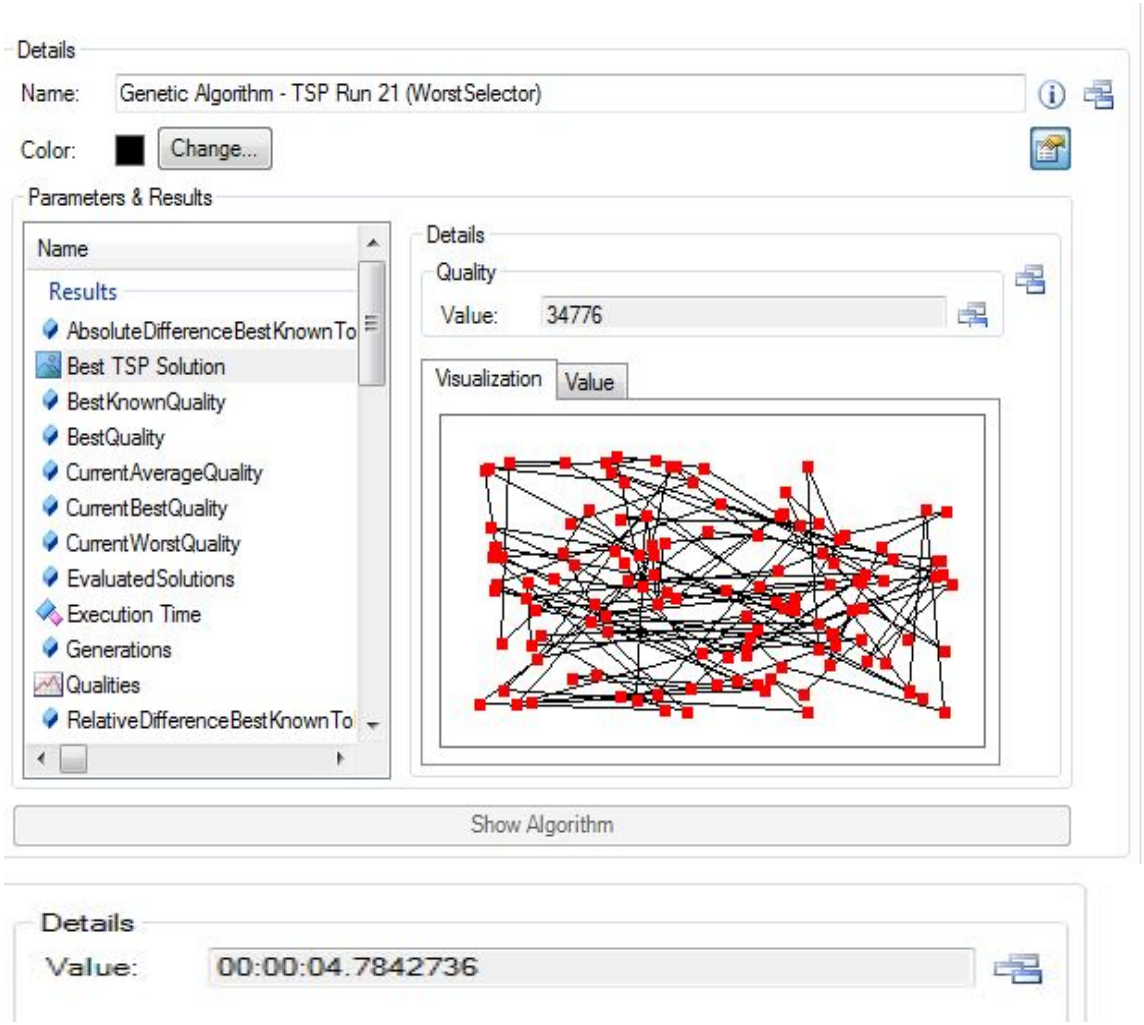
3.20 Rulare cu selector TournamentSelector

PopulationSize: 100
Elites: 1
MutationProbability: 5%
MaximumGenerations: 1000
Selector: TournamentSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



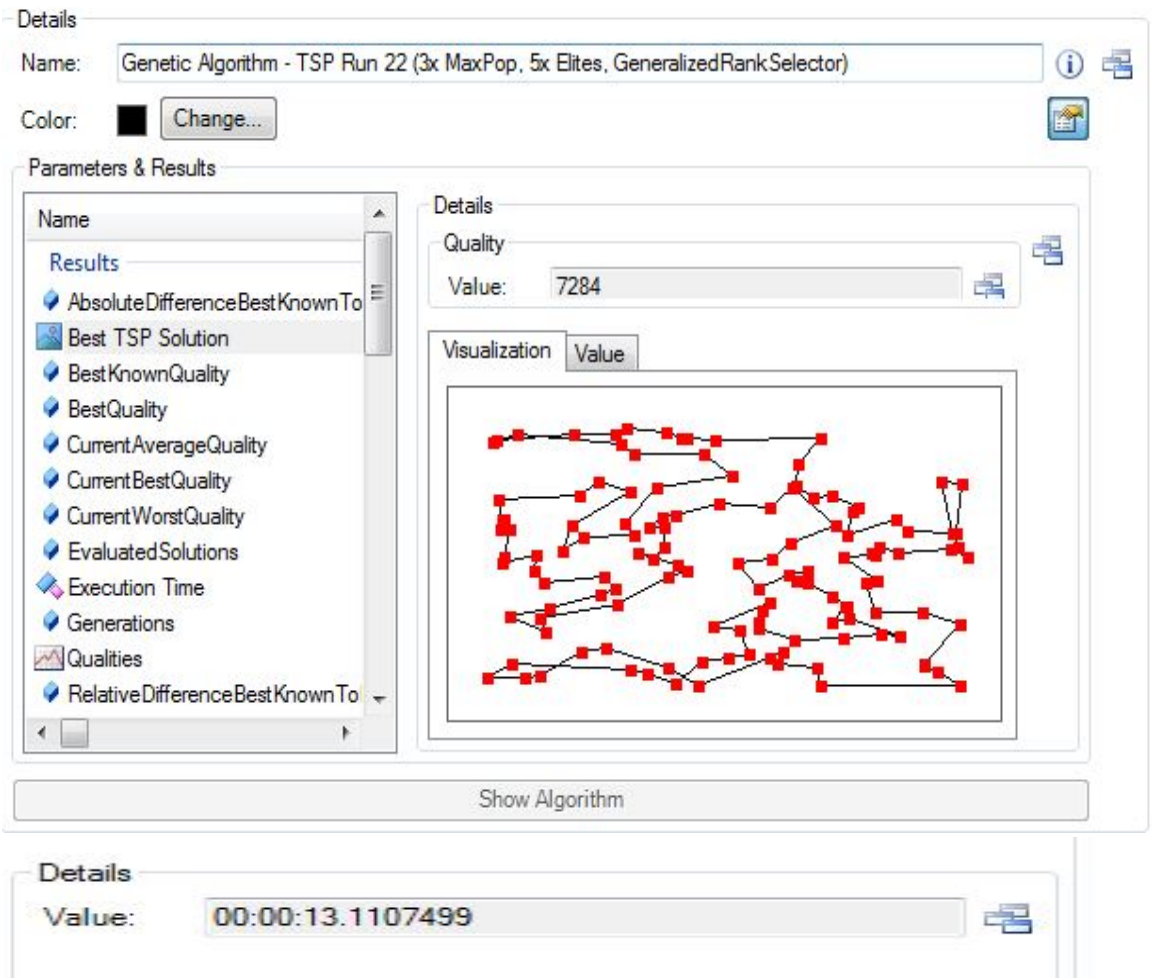
3.21 Rulare cu selector WorstSelector

PopulationSize: 100
Elites: 1
MutationProbability: 5%
MaximumGenerations: 1000
Selector: WorstSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



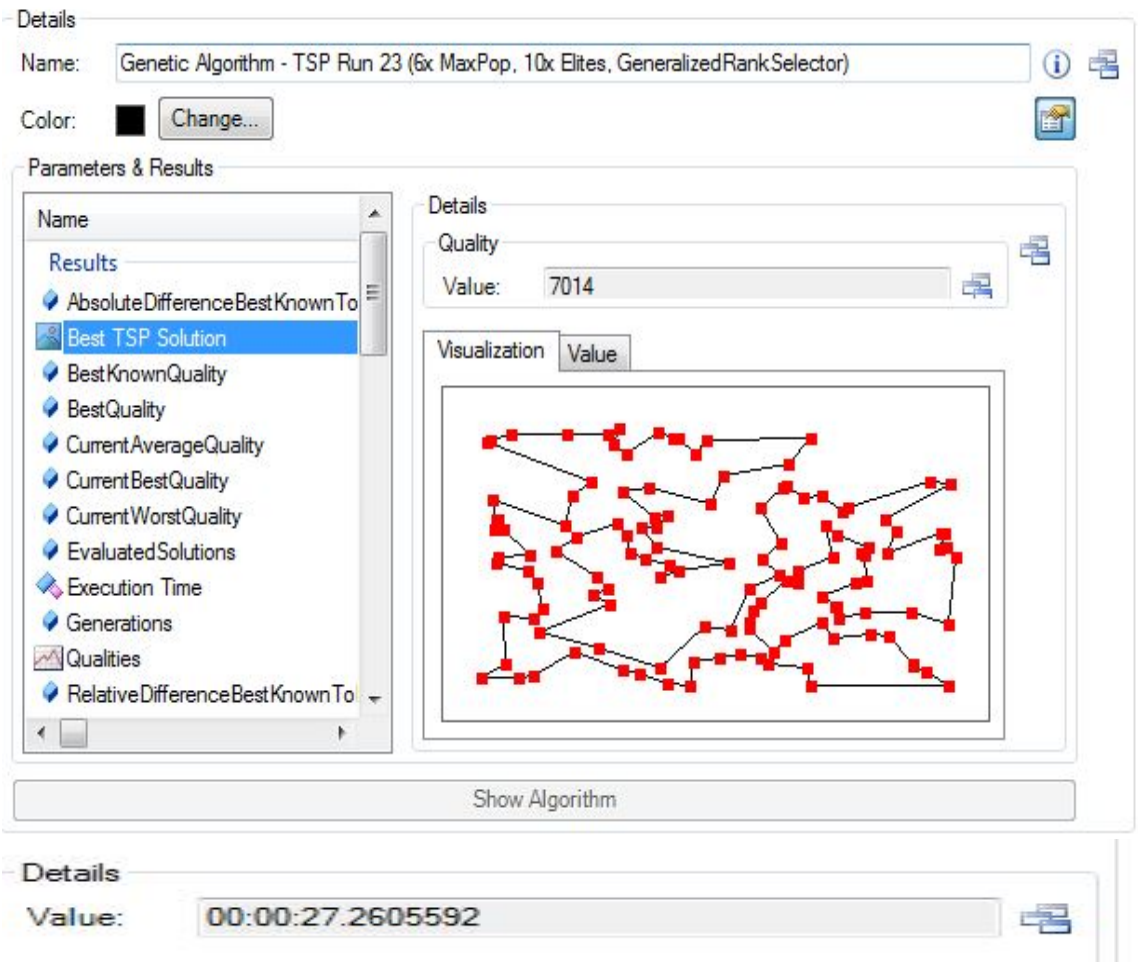
3.22 Rulare cu generație maximă triplă, 5 membrii de elită și selector GeneralizedRankSelector

PopulationSize: 100
Elites: 5
MutationProbability: 5%
MaximumGenerations: 3000
Selector: GeneralizedRankSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



3.23 Rulare cu generație maximă 6000, 10 membrii de elită și selector GeneralizedRankSelector

PopulationSize: 100
Elites: 10
MutationProbability: 5%
MaximumGenerations: 6000
Selector: GeneralizedRankSelector
Crossover: OrderCrossover2
Mutator: InversionManipulator



4. Concluzii

- Cu cât crește generația maximă, cu atât crește eficiența găsirii drumului optim, dar și durata de execuție a algoritmului;
- Cele mai bune rezultate se obțin cu o probabilitate de mutație egală cu sau în jur de 5%;
- Cu cât crește numărul membrilor de elită, cu atât crește eficiența găsirii drumului optim și scade durata de execuție a algoritmului;
- Selectorii GeneralizedRankSelector, LinearRankSelector, NoSameMatesSelector și TournamentSelector obțin rezultate mai bune decât selectorul ProportionalSelector;
- Selectorul random produce rezultate diferite de fiecare dată când este folosit;
- Rezultate optime se obțin cu o populație de 100 de indivizi per generație;
- Cel mai optim rezultat se află undeva sub 7000, datorită faptului că, pe măsură ce ne apropiem de valoarea de 7000, trebuie să modificăm într-un mod pozitiv mai mulți parametrii și să le atribuim valori care contribuie cât mai mult la schimbare într-un mod pozitiv a rezultatului ca să obținem schimbări.