Problema rucsacului

1.Descrierea problemei

Problema rucsacului este o problemă de optimizare combinatorică⁠. Dată fiind o mulțime de elemente, fiecare cu o greutate și o valoare, se determină numărul din fiecare element al mulțimii care poate fi inclus într-o colecție, astfel încât greutatea totală să fie mai mică sau egală cu o anumită limită și valoarea totală să fie cât mai mare.

2.Rezolvare prin algoritm genetic

[START] Inițializăm populația de indivizi (indivizii sunt elementele din rucsac).

[FITNESS] Prin intermediul funcției de fitness, evaluăm fiecare individ din populație (verifică colecția produselor).

[ELITE] Din populația curentă, păstrăm individul pentru care s-au obținut cele mai bune rezultate în funcția de fitness (colecția cea mai mare).

[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).

[MUTATION] În funcție de probabilitatea de mutație, există posibilitatea ca noul individ să sufere mutații (două valori 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 final.

3.Rulare în HeuristicLab

**3.1 Rulare cu parametrii de bază**

Elites: 1

MaximumGenerations: 1000

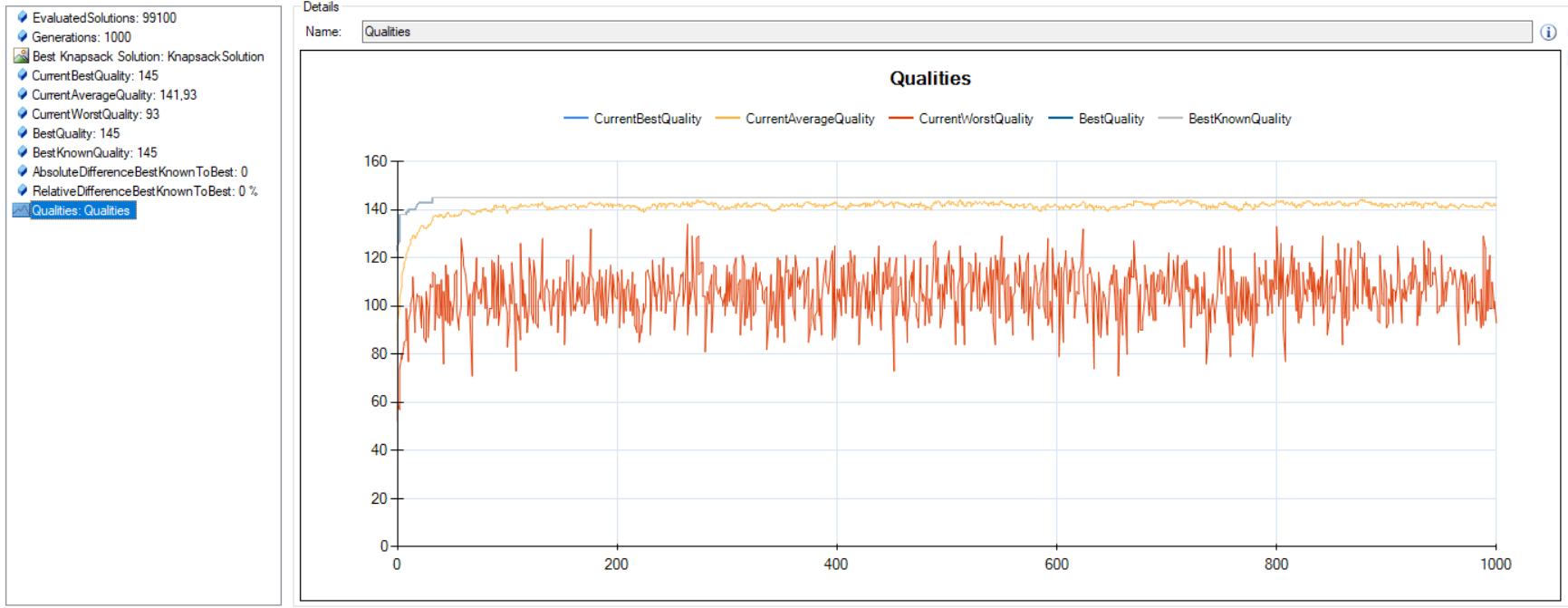
MutationProbability: 5%

PopulationSize: 100

Crossover: MultiBinaryVectorCrossover

Mutator: SomePositionsBitflipManipulator

Selector: ProportionalSelector



**3.2 Rulare cu generație maximă 10**

Elites: 1

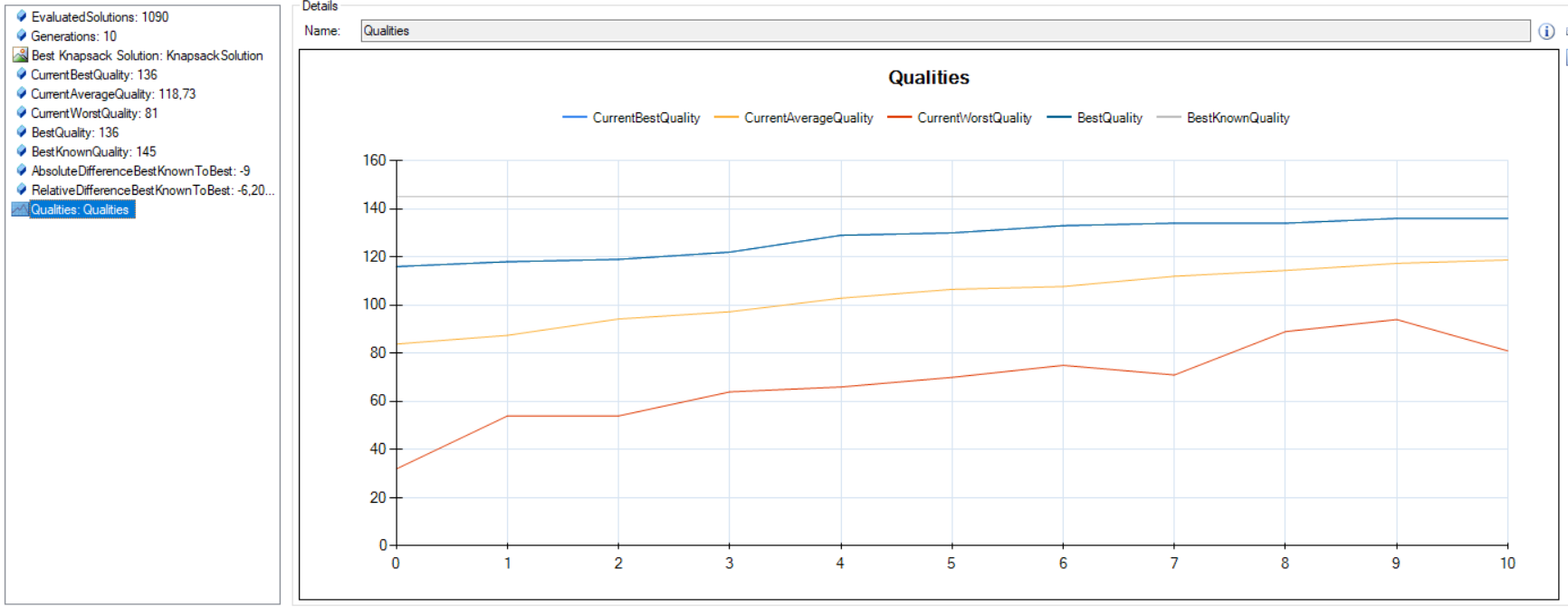
MaximumGenerations: 10

MutationProbability: 5%

PopulationSize: 100

Crossover: MultiBinaryVectorCrossover

Mutator: SomePositionsBitflipManipulator

Selector: ProportionalSelector

**3.3 Rulare cu generație maximă 100**

Elites: 1

MaximumGenerations: 100

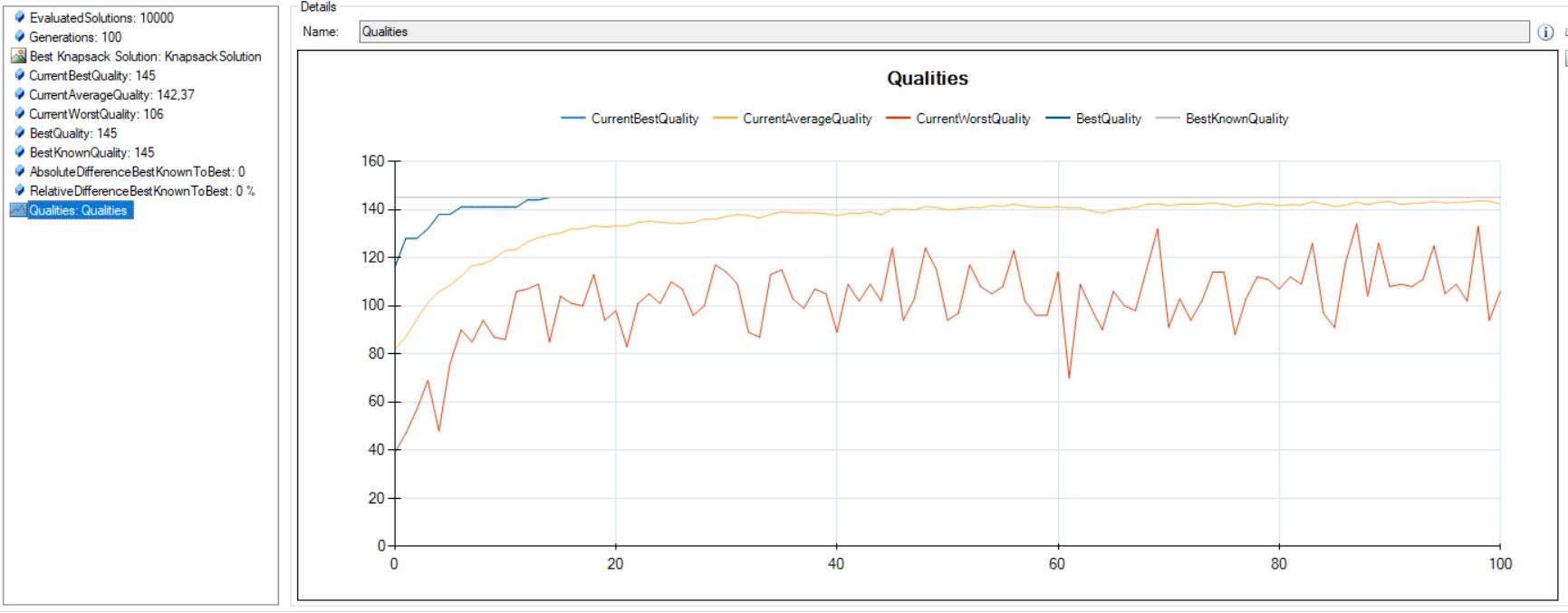
MutationProbability: 5%

PopulationSize: 100

Crossover: MultiBinaryVectorCrossover

Mutator: SomePositionsBitflipManipulator

Selector: ProportionalSelector



**3.4 Rulare cu generație maximă 5000**

Elites: 1

MaximumGenerations: 5000

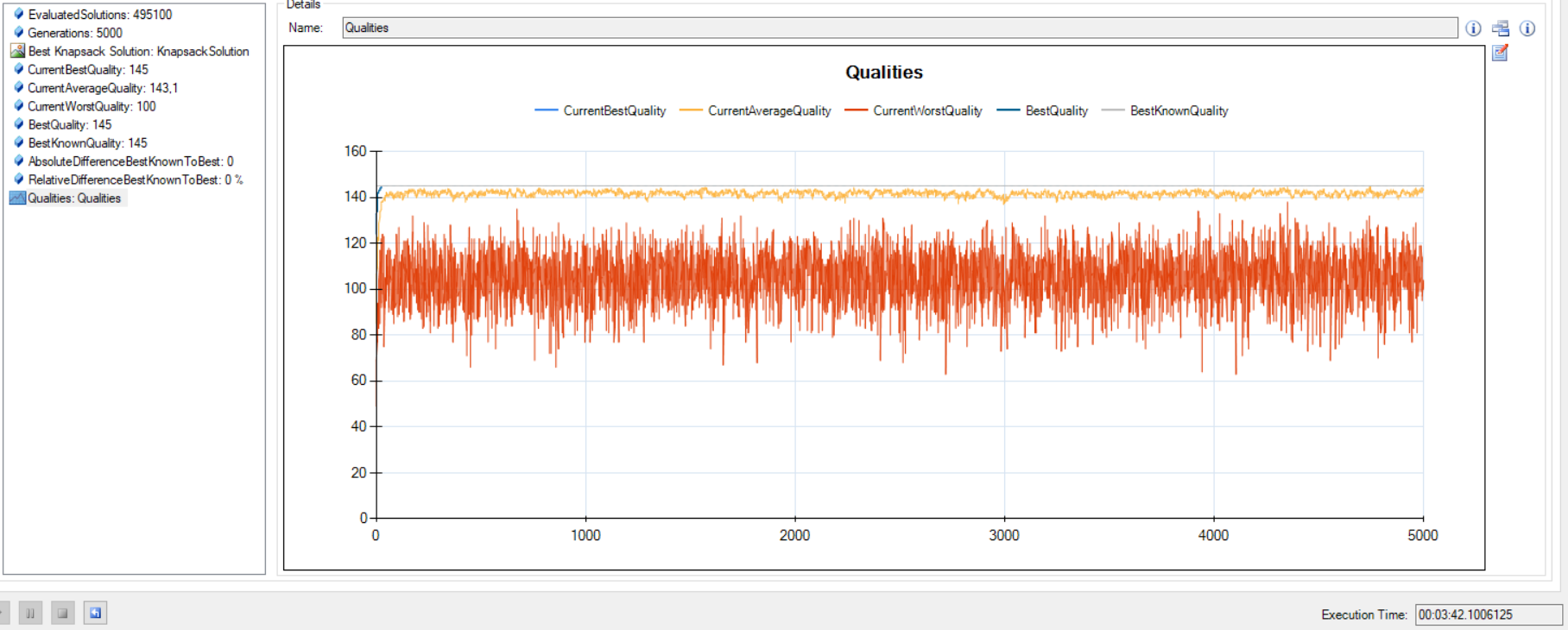
MutationProbability: 5%

PopulationSize: 100

Crossover: MultiBinaryVectorCrossover

Mutator: SomePositionsBitflipManipulator

Selector: ProportionalSelector



**3.5 Rulare cu probabilitate de mutație 1%**

Elites: 1

MaximumGenerations: 1000

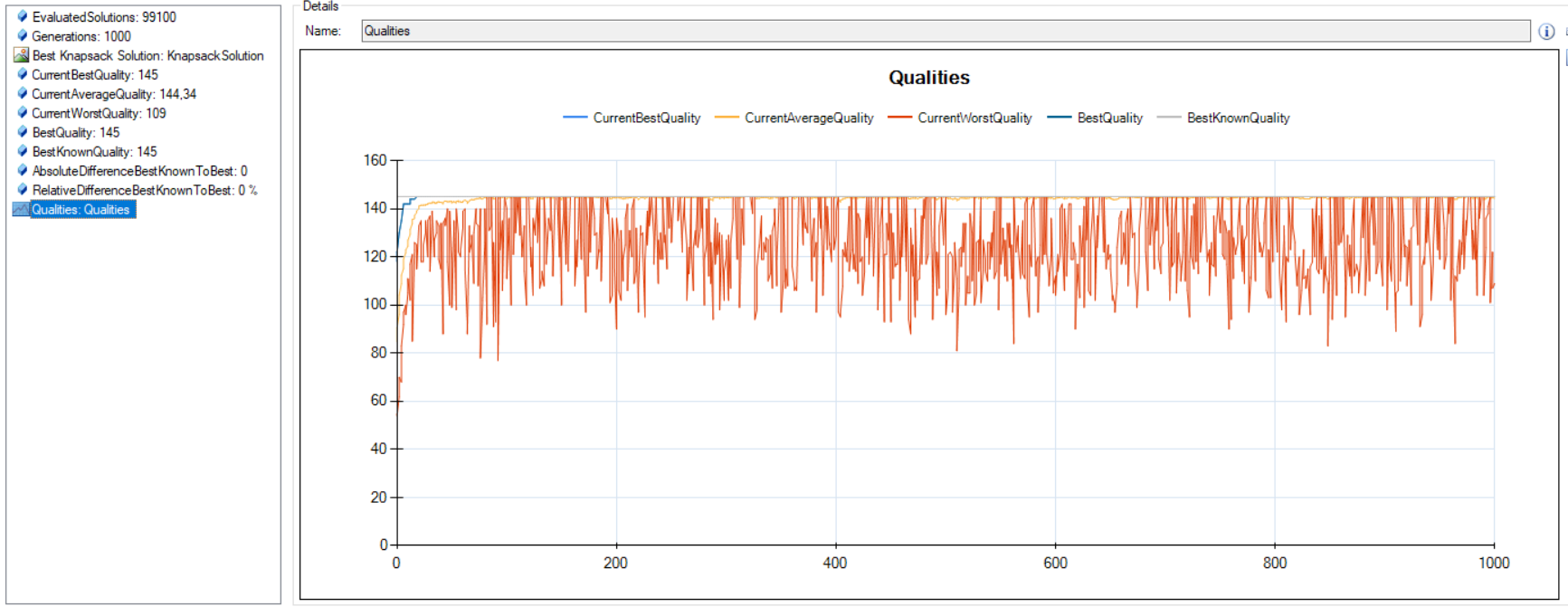
MutationProbability: 1%

PopulationSize: 100

Crossover: MultiBinaryVectorCrossover

Mutator: SomePositionsBitflipManipulator

Selector: ProportionalSelector



**3.6 Rulare cu probabilitate de mutație 50%**

Elites: 1

MaximumGenerations: 1000

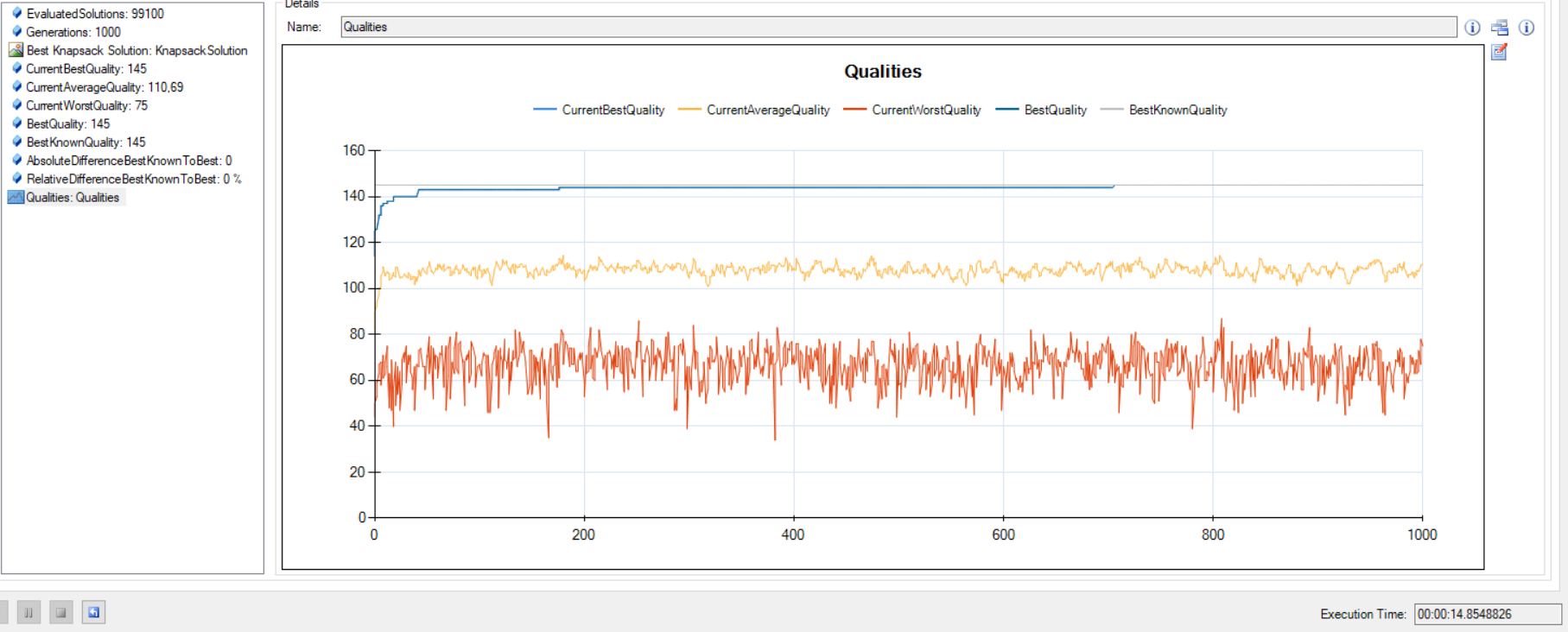
MutationProbability: 50%

PopulationSize: 100

Crossover: MultiBinaryVectorCrossover

Mutator: SomePositionsBitflipManipulator

Selector: ProportionalSelector



**3.7 Rulare cu probabilitate de mutație 100%**

Elites: 1

MaximumGenerations: 1000

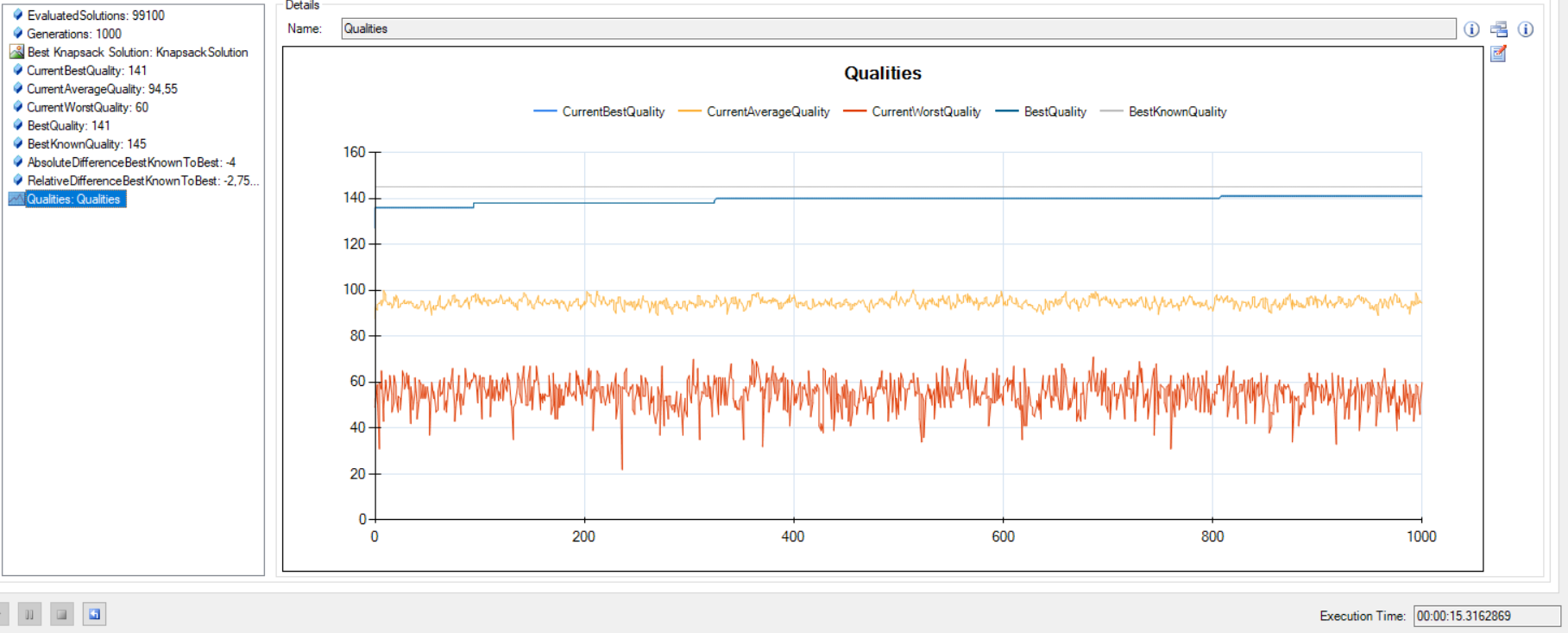
MutationProbability: 100%

PopulationSize: 100

Crossover: MultiBinaryVectorCrossover

Mutator: SomePositionsBitflipManipulator

Selector: ProportionalSelector



**3.8 Rulare cu populație maximă 20**

Elites: 1

MaximumGenerations: 1000

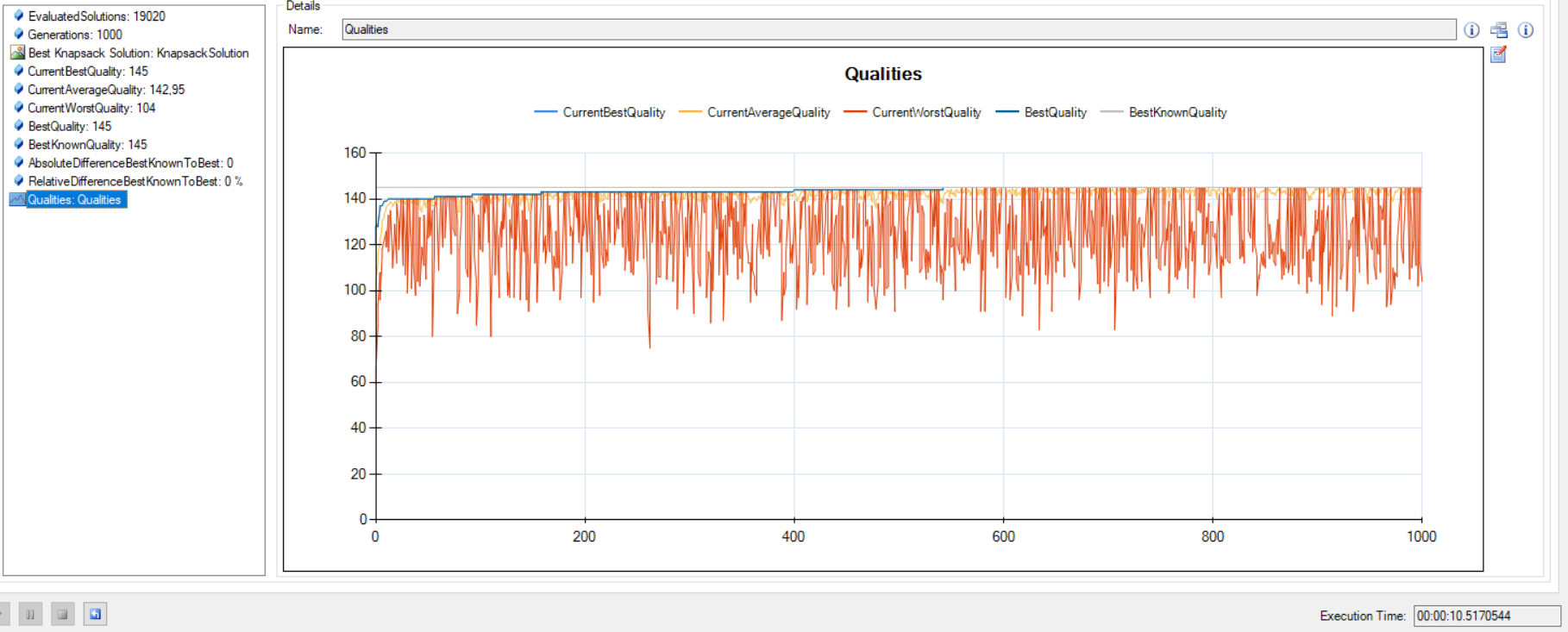
MutationProbability: 5%

PopulationSize: 10

Crossover: MultiBinaryVectorCrossover

Mutator: SomePositionsBitflipManipulator

Selector: ProportionalSelector



**3.9 Rulare cu populație maximă 500**

Elites: 1

MaximumGenerations: 500

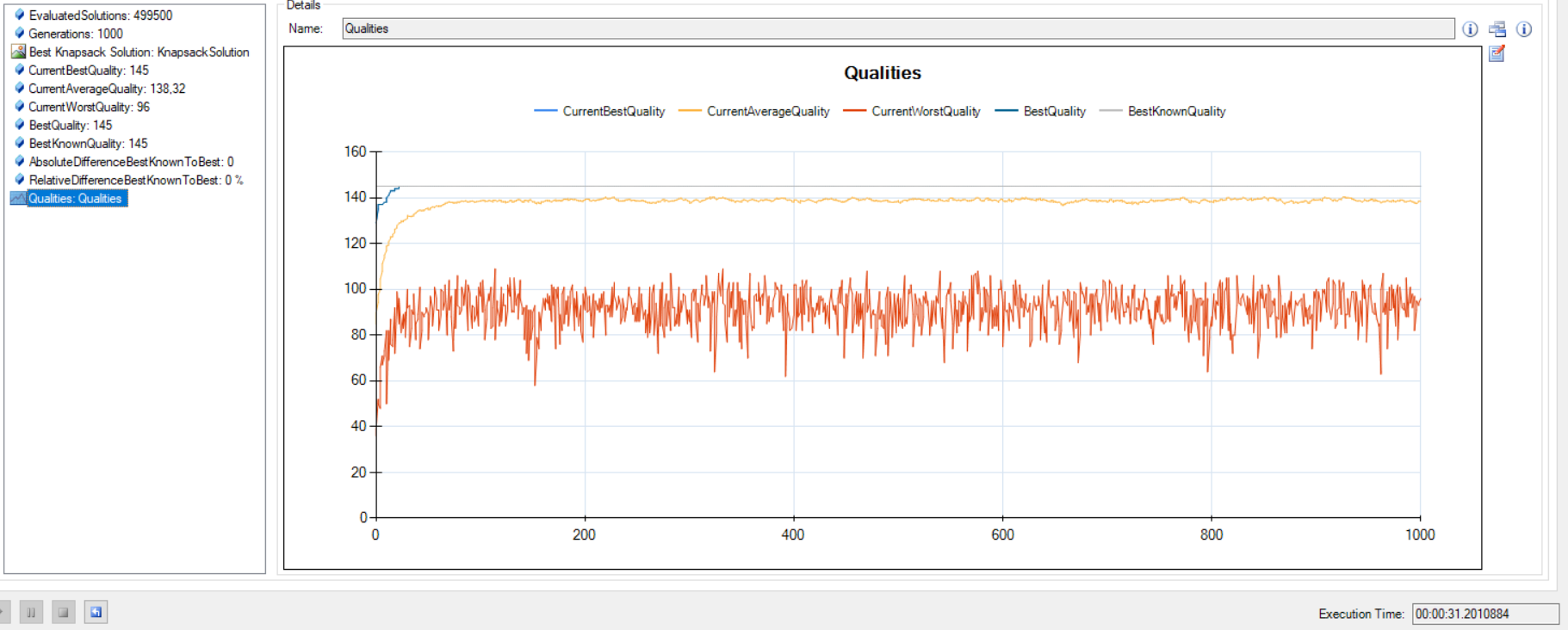
MutationProbability: 5%

PopulationSize: 1000

Crossover: MultiBinaryVectorCrossover

Mutator: SomePositionsBitflipManipulator

Selector: ProportionalSelector



**3.10 Rulare cu 5 membrii de elită**

Elites: 5

MaximumGenerations: 1000

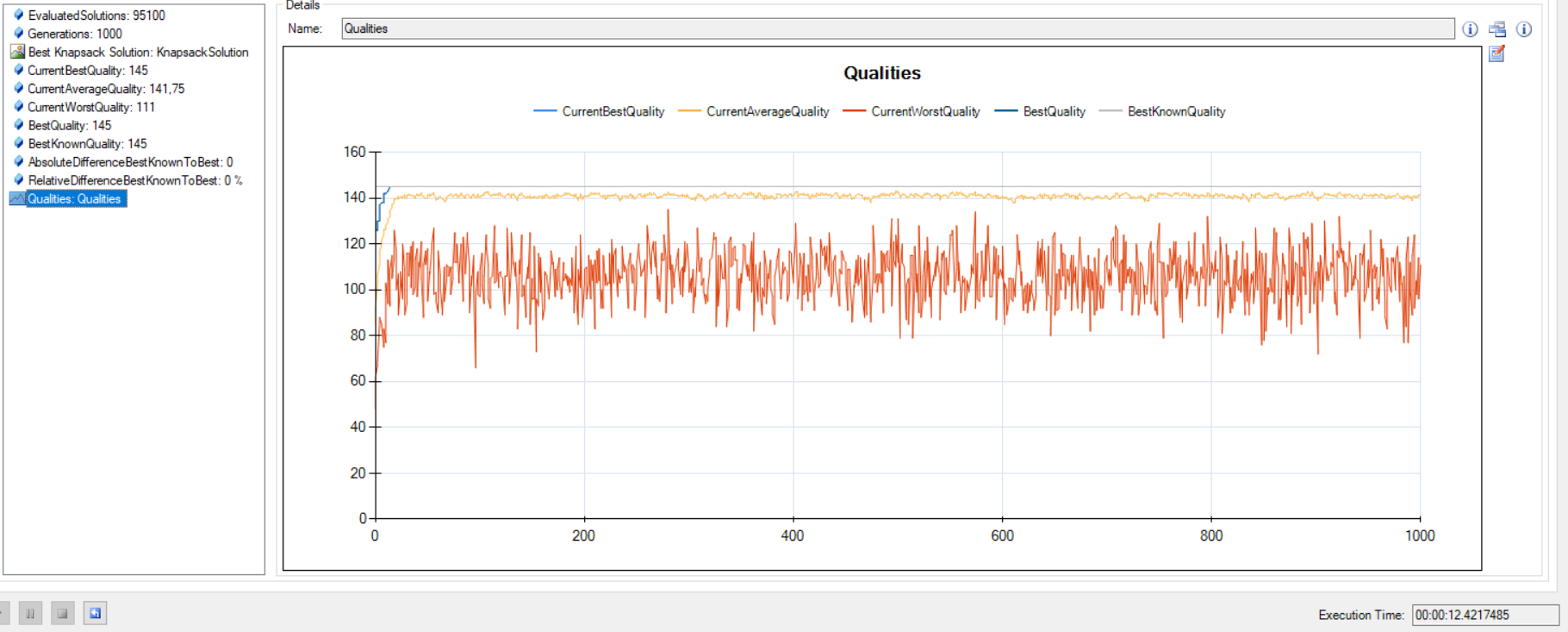
MutationProbability: 5%

PopulationSize: 100

Crossover: MultiBinaryVectorCrossover

Mutator: SomePositionsBitflipManipulator

Selector: ProportionalSelector



**3.11 Rulare cu 10 membrii de elită**

Elites: 10

MaximumGenerations: 1000

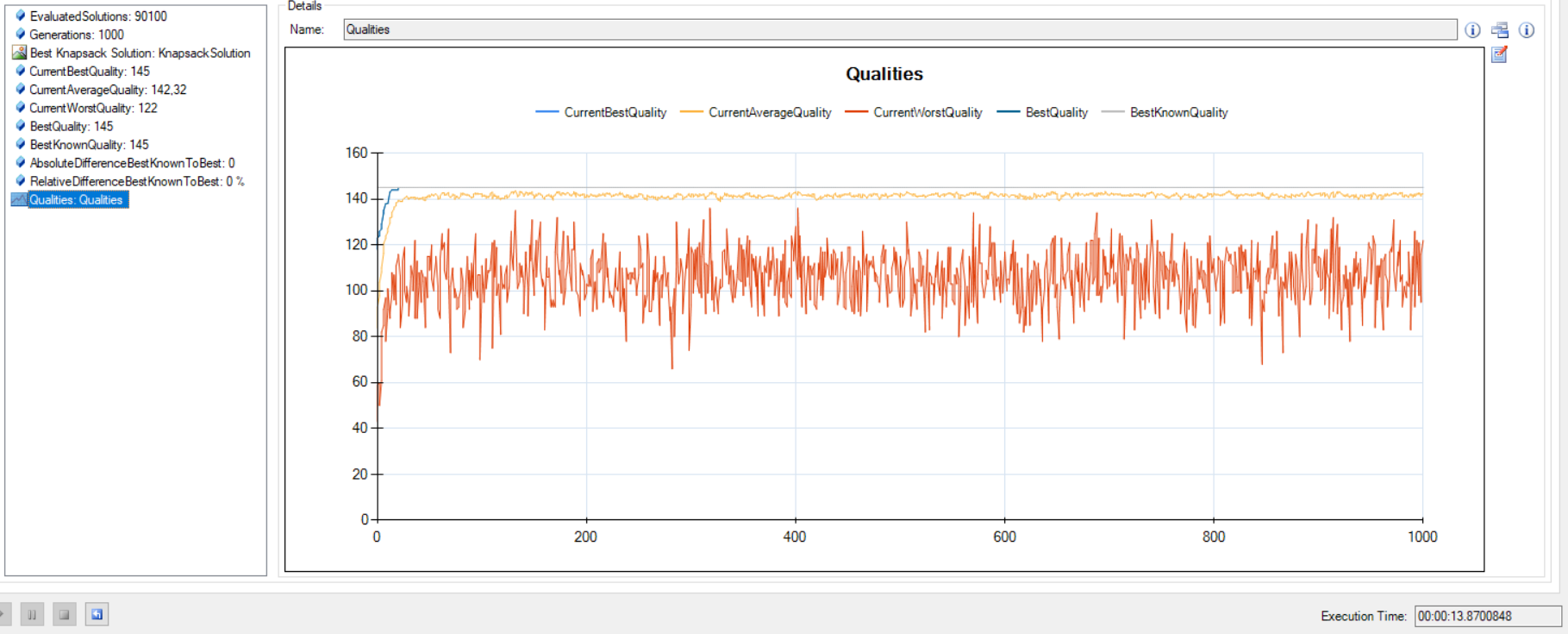
MutationProbability: 5%

PopulationSize: 100

Crossover: MultiBinaryVectorCrossover

Mutator: SomePositionsBitflipManipulator

Selector: ProportionalSelector



**3.12 Rulare cu selector LinearRankSelector**

Elites: 1

MaximumGenerations: 1000

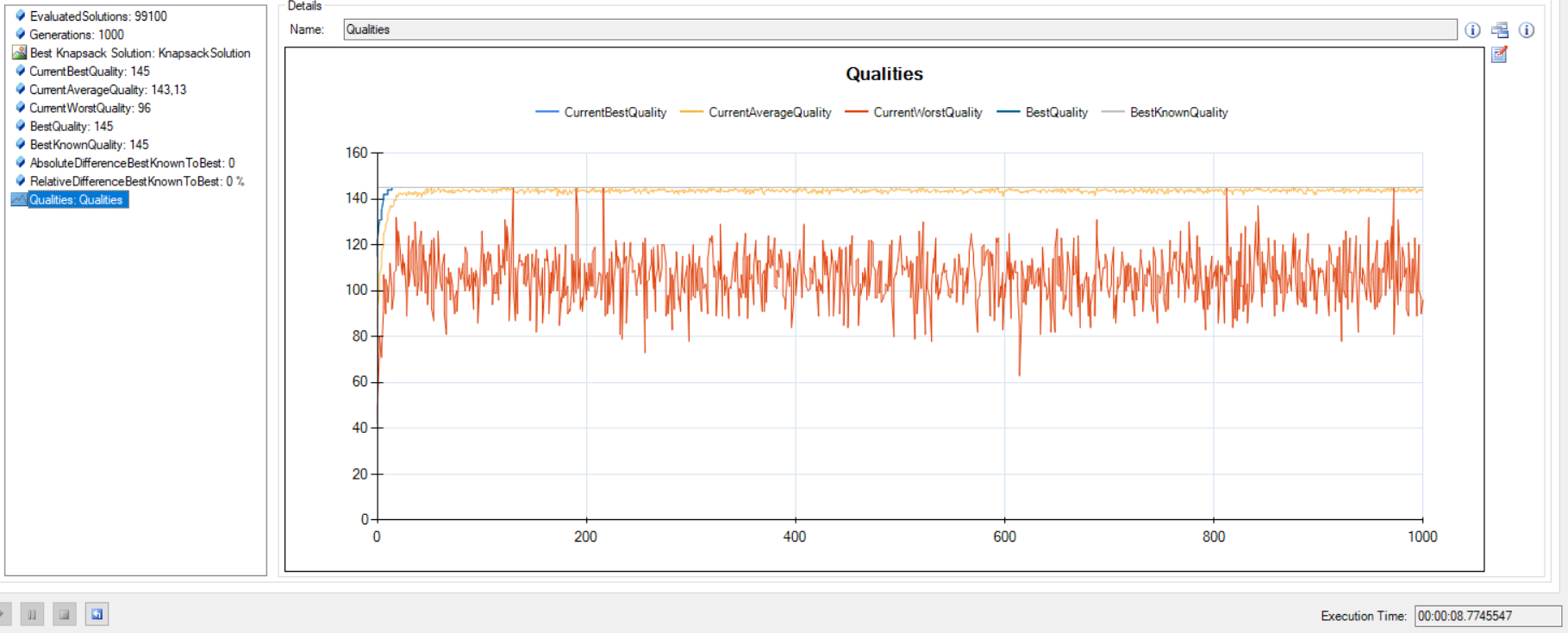
MutationProbability: 5%

PopulationSize: 100

Crossover: MultiBinaryVectorCrossover

Mutator: SomePositionsBitflipManipulator

Selector: LinearRankSelector



**3.13 Rulare cu selector WorstSelector**

Elites: 1

MaximumGenerations: 1000

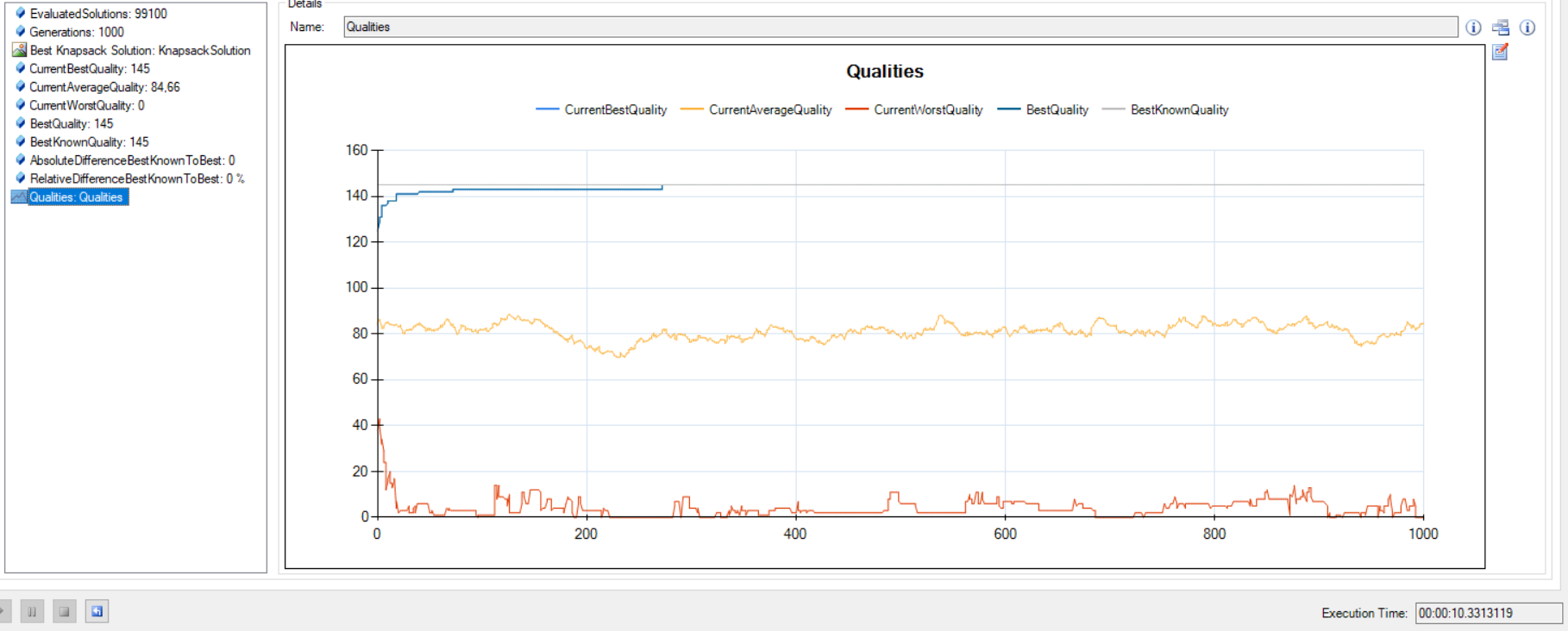
MutationProbability: 5%

PopulationSize: 100

Crossover: MultiBinaryVectorCrossover

Mutator: SomePositionsBitflipManipulator

Selector: WorstSelector



**3.14 Rulare cu generație maximă 200, probabilitate de mutatie 1% și selector** LinearRankSelector

Elites: 1

MaximumGenerations: 200

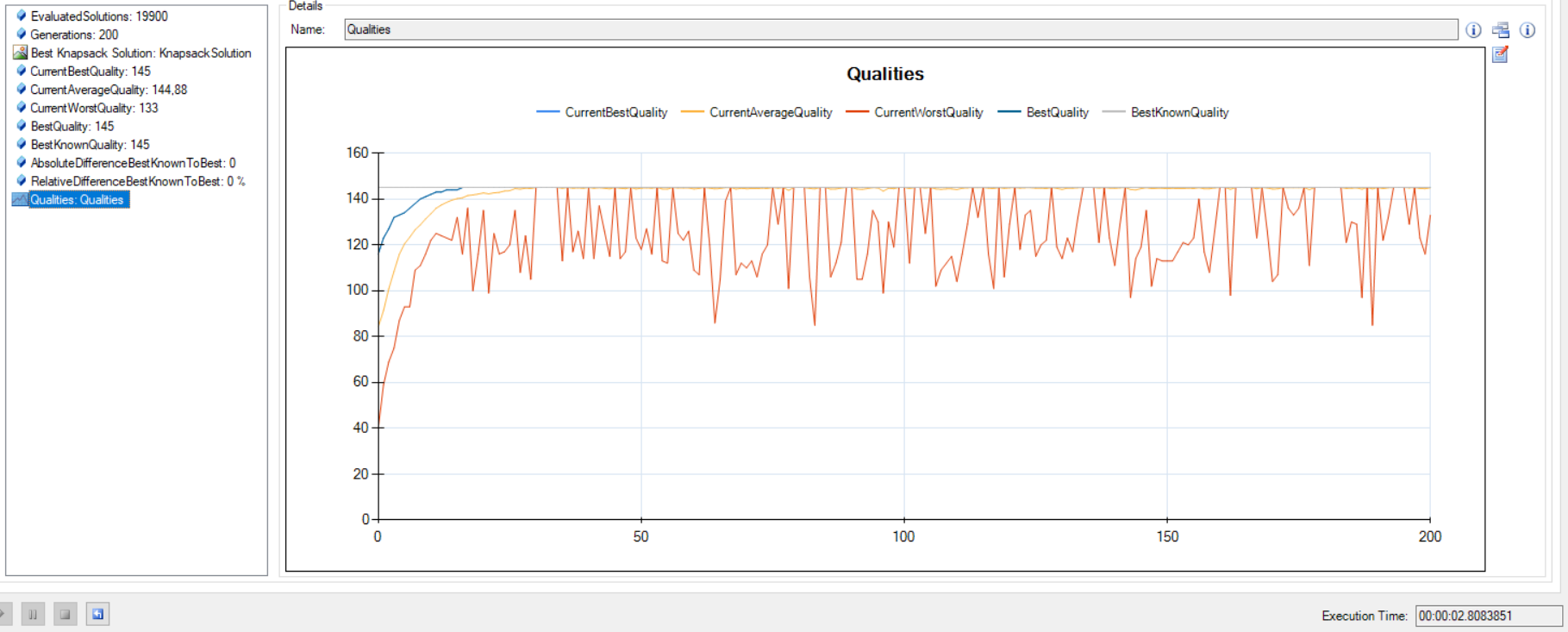
MutationProbability: 1%

PopulationSize: 100

Crossover: MultiBinaryVectorCrossover

Mutator: SomePositionsBitflipManipulator

Selector: LinearRankSelector



4. Concluzii

* Dacă generația maximă este mai mare de 100 nu influențează calitatea găsirii celui mai bune colecții.
* Dacă populația este mai mică atunci numărul de calcule este mai mare până când se găsește cea mai bună colcție.
* Numărul de elite nu influențează găsirea colecției optime.
* Cele mai neoptime colecții se obțin cu cât probabilitatea de mutație este mai mare
* Selectorii GeneralizedRankSelector, LinearRankSelector, NoSameMatesSelector și TournamentSelector obțin cele mai bune rezultate .