ParadisEO-MO:
Fitness Landscape Analysis and efficient local search implementation

29/10/2010

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

INRIA

centre de recherche LILLE - NORD EUROPE

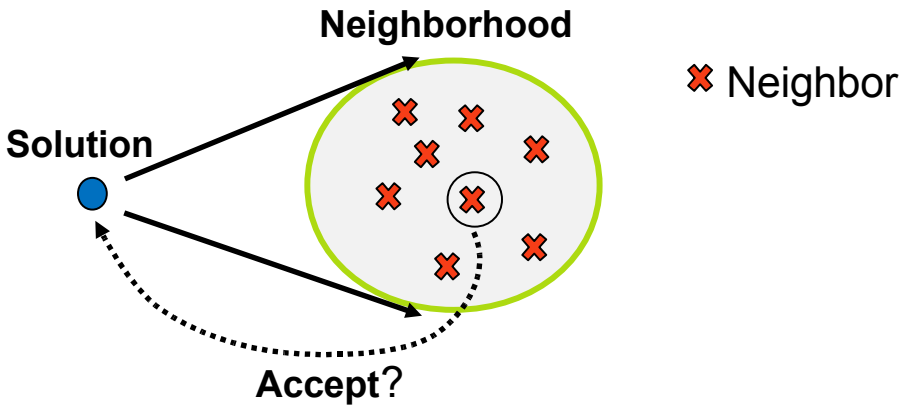
The ParadisEO Team,
including:

J. Humeau,
A. Liefooghe,
S. Verel.

Concepts in ParadisEO-MO: "Neighbor", "Neighborhood" and "Evaluation"

- "Neighbor"
  - Move + save neighbor informations (fitness and more)
- "Neighborhood"
  - Describe how to compute all the neighbors
  - Methods: init, next, cont and hasNeighbor
- "Evaluation"
  - "moEval": set fitness (and more) of the neighbor
  - Can be incremental or full evaluation

General schema of LS



Search explorer:

- Generate neighbors from the neighborhood
- Select a neighbor
- Decide to replace solution by the selected neighbor

Simplified schema of LS

Constructors:

- moNeighborhood(moNeighbor)
- moSearchExplorer(moNeighborhood, moEval)

```
do {
        searchExplorer(solution)
        if ( searchExplorer.accept(solution) )
                searchExplorer.move(solution)
}
while (searchExplorer.continue(solution))
```

Completed schema of LS

```
searchExplorer.initParam(solution)
continuator.init(solution)
do {
        searchExplorer(solution)
        if ( searchExplorer.accept(solution) )
                searchExplorer.move(solution)
        searchExplorer.updateParam(solution)
}
while (continuator(solution) && searchExplorer.continue(solution))
searchExplorer.terminate(solution)
```

Statistics can be computed (and reported):

- at initialization: continuator.init(solution)
- at each iteration : continuator(solution)

Improvements

- Hill-climbing like:
  - Simple HC (best improvement)
  - Simple HC neutral (random choice among several best solutions) NEW
  - HC neutral (+ allows move when best solution has equal fitness) NEW
  - First Improvement HC
- Walk like to sample search space: NEW
  - Random walk
  - Random neutral walk
  - Metropolis hasting
- Simulated Annealing

  cooling schudeling: init, decreases, final NEW
- Tabu Search

  "*moMemory" : diversification, intensification, tabu "list"* NEW

Classical LS: improvements

- Hill-climbing like:
  - Simple HC (best improvement)
  - Simple HC neutral (random choice among several best solutions) NEW
  - HC neutral (+ allows move when best solution has equal fitness) NEW
  - First Improvement HC
- Walk like to sample search space: NEW
  - Random walk
  - Random neutral walk
  - Metropolis hasting
- Simulated Annealing
  cooling schudeling: init, decreases, final NEW
- Tabu Search
  "moMemory" : diversification, intensification, tabu "list" NEW

Modularity improved
Easy reuse of basic LS

Statistics and checkpointing : Fitness Landscape Analysis

NEW

- Checkpointing like in ParadisEO-EO is available
  - moCheckpoint
  - moContinuator
  - moStatBase
    - Statistics in the Neighborhood : min, max, mean, standard deviation, probability to increase, neutral degree…
    - Autocorrelation of fitness, fitness cloud, FDC…

Fitness Landscape Analysis NEW

Class to define a sampling:

- moSampling<Neighbor> (*paradiseo-mo/src/sampling)*

Constructor:

```
moSampling (eoInit< EOT > &_init,
            moLocalSearch< Neighbor > &_localSearch,
            moStat< EOT, ValueType > &_stat,
            bool _monitoring=true)
```

Method:

```
add(moStat< EOT, ValueType > &_stat, bool _monitoring=true)
```

NEW

Available Statistics

- moFitnessStat : the fitness of the current solution
- moDistanceStat : the distance between the current solution and a given solution
- moSolutionStat : the current solution
- moCounterStat : the number of iterations
- moBestSoFarStat : the best current solution found
- moNeighborBestStat : best fitness over k neighbors
- moNeighborhoodStat : to compute the statistics from the neighbors solutions : .....

NEW

Available Statistics

- moNeighborhoodStat : to compute the statistics from the neighbors solutions :
  - moAverageFitnessNeighborStat : average fitness in the neighborhood
  - moStdFitnessNeighborStat : standard deviation of fitness
  - moMaxNeighborStat : maximum fitness
  - moMinNeighborStat : minimum fitness
  - moSecondMomentNeighborStat : average and standard deviation
  - moSizeNeighborStat : size of the neighborhood
  - moNeutralDegreeNeighborStat : number of neighbors with equal fitness
  - moNbInfNeighborStat : number of neighbors with lower fitness
  - moNbSupNeighborStat : number of neighbor with higher fitness

Fitness Landscape Analysis

Pre-defined classes NEW

- moDensityOfStatesSampling : density of states
- moAutocorrelationSampling : autocorrelation length and functions
- moFDCsampling : fitness distance correlation
- moHillClimberSampling : adaptive walks
- moNeutralDegreeSampling : neutral degree
- moNeutralWalkSampling : evolvability of neutral networks
- moFitnessCloudSampling : evolvability of the operator, and the neighborhood

(In directory *paradiseo-mo/src/sampling)*

How to use ParadisEO-MO? What I need define?

Only problem dependent components:

- Solution representation *EOT*
- "Random" initialisation operator *eoInit*
- Evaluation Functions
  - full Evaluation *eoEvalFunc*
  - incremental (if possible) *moEval*
- Neighbor *moNeighbor*
- Neighborhood
  - *moNeighborhood*
  - or a mapping (int -> Neighbor) using an *moIndexNeighborhood* NEW

Predefined components for the classical search spaces: NEW

- Bit string
- Permutation

Example of code (from tutorial lesson 6)

```
eoUniformGenerator<bool> uGen;
eoInitFixedLength<Indi> random(vecSize, uGen);

oneMaxFullEval<Indi> fullEval;

eoHammingDistance<Indi> distance;

Indi bestSolution(vecSize, true); // global optimum

moFDCsampling<Neighbor> sampling(random, fullEval, distance, bestSolution, nbSol);

sampling();

sampling.fileExport(str_out);

const std::vector<double> & fitnessValues = sampling.getValues(0);
const std::vector<double> & distValues    = sampling.getValues(1);
```

Conclusion

- More reusable design (“Neighbor”, “Neighborhood”, …)
- 11 stantard local search vs 6 before
- More than 133 classes (≈ 13500 code lines) vs 40 before
- All classes are tested and documentation available
- fitness landscape analysis is now available
- Standard components added (Bit string, permutation and so…)
- …A lot of NEW

Perspectives

- Create tool boxes for automatics fitness landscape analysis
- Extend to multiobjective optimization: set-based local search
- Create problems repository (done in fact)
- Create tools for automatics parameters tunings and parameters control
- Connection with GPGPU