LOCAL SEARCH METHODS
APPLICATIONS AND ENGINEERING

Lecture 6

Application Examples and Timetabling

Marco Chiarandini

Outline

1. Metaheuristics (continued)

2. Application Examples

Outline

1. Metaheuristics (continued)

2. Application Examples

Evolutionary Computation Algorithms (continued)

The lingo of Evolutionary Computation:
correspondence between artificial elements and natural counterparts

- strings $\cong$ chromosomes
- solution set $\cong$ genotype
- candidate solution $\cong$ phenotype
- solution components $\cong$ genes
- values for solution components $\cong$ alleles
- positions in the string $\cong$ loci
- objective function $\cong$ fitness
- dependencies between string positions $\cong$ epistasis

Replacement

Offsprings can either replace the whole population (generational approach) or replace less fit individuals (steady-state approach)

Memetic algorithm

Lamarckian *vs*Darwinian approach

Scatter Search and Path Relinking

Key idea: maintain a small population of *reference solutions* and combine them to create new solutions.

Differ from EC by providing unified principles for recombining solutions based on generalized path constructions in Euclidean or neighborhood spaces.

Scatter Search and Path Relinking:

generate sp with a *diversification generation method*
perform *subsidiary perturbative search* on sp
extract reference set rs from sp
While *termination criterion* is not satisfied:
  generate subset sc from rs
  generate solution s from sc by a *combination operator*
  perform *subsidiary perturbative search* on s
  let s_{worst} be the worst solution in rs
  if $s \notin rs$ and $g(s) < g(s_{worst})$
    substitute s_{worst} with s in rs

Note:

- A large number of solutions is generated by the *diversification generation* method while about $1/10$ of them are chosen for the *reference set*.
- In more complex implementations the size of the subset of solutions sc may be larger than two.

Scatter Search

Solutions are encoded as points of an Euclidean space and
new solutions are created by building linear combinations of reference solutions
using both positive and negative coefficients.

Path Relinking

Combinations are reinterpreted as paths between solutions in a neighborhood space. Starting from an *initiating solution* moves are performed that introduces components of a *guiding solution*.

Cross Entropy Method

Key idea: use *rare event-simulation* and *importance sampling* to proceed toward good solutions

- Generate random solution samples according to a specified mechanism
- update the parameters of the random mechanism to produce better "sample"

Cross Entropy Method (CEM):

Define $\hat{\mathbf{v}}_0 = \mathbf{u}$. Set $t = 1$
While *termination criterion* is not satisfied:
  generate a sample $(s_1, s_2, \dots s_N)$ from the pdf $p(\cdot; v_{t-1})$
  set $\hat{\gamma}_t$ equal to the $(1-\rho)$-quantile with respect to g
  use the same sample $(s_1, s_2, \dots, s_N)$ to solve the stochastic program

$$\hat{\mathbf{v}}_t = \arg\max_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^{N} I_{\{g(s_i) \leq \hat{\gamma}_t\}} \ln p(s_i; \mathbf{v})$$

Generates a two-phase iterative approach to construct a sequence of levels $\hat{\gamma}_1, \hat{\gamma}_2, \dots, \hat{\gamma}_t$ and parameters $\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \dots, \hat{\mathbf{v}}_t$ such that $\hat{\gamma}_t$ is close to optimal and $\hat{\mathbf{v}}_t$ assigns maximal probability to sample high quality solutions

Termination criterion: if for some $t \geq d$ with, *e.g.*, $d = 5$,
$$\hat{\gamma}_t = \hat{\gamma}_{t-1} = \ldots = \hat{\gamma}_{t-d}$$
Smoothed Updating: $\hat{\mathbf{v}}_t = \alpha\hat{\mathbf{v}}_t + (1-\alpha)\hat{\mathbf{v}}_{t-1}$ with $0.4 \leq \alpha \leq 0.9$
Parameters: $N = cn$, $c > 1$ $(5 \leq c \leq 10)$; $\rho \approx 0.01$ for $n \geq 100$ and $\rho \approx \ln(n)/n$ for $n < 100$

Example: TSP

- Solution representation: permutation representation
- Probabilistic model: matrix P where p_{ij} represents probability of vertex j after vertex i
- Tour construction: specific for tours

  Define $P^{(1)} = P$ and $X_1 = 1$. Let $k = 1$
  While $k < n - 1$
    obtain $P^{(k+1)}$ from $P^{(k)}$ by setting the X_k-th column of $P^(k)$ to zero and normalizing the rows to sum up to 1.
    Generate X_{k+1} from the distribution formed by the X_k-th row of $P^{(k)}$
    set $k = k + 1$

- Update: take the fraction of times transition i to j occurred in those paths the cycles that have $g(s) \leq \gamma$

Estimation of Distribution Algorithms

Key idea avoid the problem of breaking good building blocks of EC by estimating a probability distribution over the search space which is then used to sample new solutions

- Candidate solutions constructed by a parametrized probabilistic model
- The candidate solutions are used to modify the model in order to bias toward high quality solutions

Needed:

- A probabilistic model
- An update rule for the model's parameter and/or structure

Estimation of Distribution Algorithm (EDA):

generate an initial population sp
While *termination criterion* is not satisfied:
  select sc from sp
  estimate the probability distribution $p_i(x_i)$ of solution component i from the highest quality solutions of sc
  generate a new sp by sampling according to $p_i(x_i)$

Probabilistic Models

No Interaction

- weighted frequencies over the population (a mutation operator can be applied to the probability)
- classical selection procedures
- incremental learning with binary strings: $p_{t+1,i}(x_i) = (1-\rho)p_{t,i}(x_i) + \rho x_i$ with $x_i \in S_{best}$

Pairwise Interaction

- chain distribution of neighboring variables (conditional probabilities constructed using sample frequencies)
- dependency tree
- forest

Multivariate

- independent clusters based on minimum description length
- factorized distribution with prior knowledge
- Bayesian optimization: Bayesian networks learning

Classification of Metaheuristics

- Trajectory methods *vs* discontinuous methods
- Population-based *vs* single-point search
- Memory usage *vs* memory-less methods
- One *vs* various neighborhood structures
- Dynamic *vs* static objective function
- Nature-inspired *vs* non-nature inspiration
- Instance based *vs* probabilistic modeling based

Outline

LS Algorithms for GCP

The algorithms and their main characteristics

- TS with complete and partial colouring approaches: prohibition rules
- Novelty algorithm (an example of randomised iterative improvement): decision tree
- GLS: weights on edges
- ILS: perturbation given by partial destruction and reconstruction
- MA: GPX crossover
- SA: random Kempe chains neighborhood

For details see Chiarandini, Dumitrescu and Stützle 2005.

LS Algorithms for SMTWTP

The algorithms and their main characteristics

- Iterated Dynamic Search (ILS): perturbation given by random interchange moves
- ACO: pheromone associted to assignment job/position heuristic value construction process pheromone update

For details see text book Chapter 9.

LS Algorithms for TSP

The algorithms and their main characteristics

- ILS perturbations acceptance criterion don't look bit issue
- MA crossovers and replacement
- $\mathcal{MAX}$-$\mathcal{MIN}$ AS pheromone update

For details see text book Chapter 8.

Guidelines for the Application of LS Methods

Perturbative Search

- The efficiency of perturbative search depends on the modeling, tuning is less crucial
- Candidate solutions must be easy to generate. In case relax constraints.
- The search space should be connected.
- The search landscape induced by the evaluation function should not be too flat. In case add components to g

Metaheuristics

- Performance hard to forecast $\Rightarrow$ implement more than one and compare
- ILS is probably the easiest and hence the first to try
- TS, DLS, PII are good to intensify the search around local optima
  Among these: TS works well when a best improvement strategy is feasible
- SA is appealing to cope with large neighborhoods and perform well when long run times are available. Tuning is crucial and good starting solutions seem to help.
- In EA pertinent information should be transmitted during the co-operation phase.
- ACO and EA perform better with pertubative search. Should be applied after the previous methods have been exploited.
- Keep it simple:
  prefer less parameters, higher degree of heuristic guidance, less components
- Hybridisations with exact methods (*e.g.*, network flow) are promising