An Improved Generic BET-AND-RUN Strategy with Performance Prediction for Stochastic Local Search

Thomas Weise (tweise@hfuu.edu.cn), Zijun Wu, Institute of Applied Optimization (IAO), Hefei University, Hefei, Anhui, China
Markus Wagner (markus.wagner@adelaide.edu.au), Optimisation and Logistics, The University of Adelaide, Adelaide, Australia

Goal
- Domain: Solving Optimization Problems
- Goal: Using a given algorithm $A$, get best possible result within time budget $t$
- very few assumptions:
  - $A$ is an iterative algorithm which attempts to improve its approximation quality over time
  - a run of $A$ can be started, paused, and resumed
  - when executing an independent run of $A$, we can get notified whenever it improves its approximation quality

Basic Bet-and-Run [1,2]
- parameters: initialization budget $0\leq t_1\leq t$, number $k\geq 0$ of initial runs
  1. start $k$ runs of $A$ and pause each of them after $t_1/k$ time units
  2. let the run with the current best approximation quality continue for the $t-t_1$ remaining time units

Generalized Bet-and-Run [5]
- parameters: initialization budget $0\leq t_1\leq t$, distribution policy $P$ (Luby [3] or even) number $m\geq k$ of initial runs, decision maker $D$, number $0<m<k$ of runs to continue
  1. start $k$ runs of $A$ and distribute the initial budget $t_1$ according to distribution policy $P$ among them
  2. apply a decision maker $D$ to choose $m$ of the $k$ runs to continue, thereby consuming $m=t_2$ time units
  3. distributed the remaining $t-t_1-t_2$ time units evenly among the $m$ selected runs

Data used for our Case Studies / Simulated Experiments
1. Minimum Vertex Cover problem (MVC)
   - algorithm: FASTVC [4]
   - data: 10,000 independent runs on each of the 86 instances from [4] generated by [5]
2. Traveling Salesperson Problem (TSP)
   - data: 10,000 independent runs on 110 symmetric instances from TSPLib + 3 additional large instances, generated by [5]
   - Potential: between 3% to 23% chance that “currentBest” decision maker could theoretically be outperformed in these datasets

Decision Makers $D$
- Idea: predict future performance of run based on last few measured (time, quality) tuples
  1. $\text{PER}(n)$: perceptron with/out single hidden layer with $n$ from 0,1,2,3
  2. polynomials (linear, quadratic, cubic)
  3. currentBest (original Bet-and-Run), currentWorst, random
  4. most or latest improvements (mostImprovements, logTimeSum)

5. diminishing: assumes that time between improvements increases exponentially, improvements decrease exponentially

Results
- currentBest is hard to beat, as beating it is impossible in most of the experiments and another method then needs to be as same as good while actually being better in the few cases where it is possible...
- PER-based decision makers can win against single runs slightly more often than currentBest on MVC and sometimes on TSP
- diminishing: simple yet surprisingly good decision maker

References