Goals
• Propose a dependency-tree EDA (dtEDA) for problems defined over permutations.
• Test dtEDA
  • Deceptive ordering problems.
  • Quadratic assignment problem (QAP).
• Compare with a GA (LOX and PMX crossover) and robust tabu search for QAP.

Background
• Two ways to represent permutations
  • Random keys (indirect)
    • Candidate solution is a vector of real numbers.
    • Sorting the numbers gives a permutation (i-th vector element’s final position is the i-th permutation element).
  • Permutation vectors (direct)
    • Directly encode permutations.
    • Must ensure that new solutions are valid permutations.
• Three types of regularities to encode in EDA models
  • Relative ordering constraints (2 goes after 5)
  • Absolute ordering constraints (5 takes 7th position)
  • Neighbor relations (2 follows 5)
• EDAs based on random keys
  • Bosman & Thierens, 2001b; Bosman & Thierens, 2001a; Robles, de Miguel, & Larrañaga, 2001
  • Can use any EDA for real-valued vectors with random keys.
  • Problems: Redundancy, inefficiency.
• EDAs with the direct representation of permutations
  • Bengoetxea et al., 2000; Tsutsui, Goldberg, & Pelikan, 2002; Zhang et al. 2003; Tsutsui, Pelikan, & Goldberg, 2006
  • Can use discrete EDAs.
  • Must introduce repair operator or modify sampling to generate only valid permutations (e.g. reject invalids).
  • Most EDAs modify sampling (relatively straightforward) and should work better (if the model is correct).
• Two more tricks
  • Sampling only subsets of permutation elements often helps.
  • Model information can be used in another crossover.

Motivation
• Estimation of distribution algorithms (EDAs) replace crossover+mutation by building and sampling an explicit probabilistic model of promising solutions.

• Successful in a number of areas from discrete to real-valued representations.
• Not as much success in permutation-based problems.

Dependency-tree EDA (dtEDA) for permutations
Pseudocode of dtEDA
• Evolve a population of permutations.
• Initial population generated at random.
• Iterate
  • Select promising permutations (e.g. tournament selection).
  • Build a dependency tree for selected permutations.
  • Sample new solutions from the built tree.
  • Replace the old solutions with the new ones.
• Output the best solution found.

Comparison on deceptive ordering problems

Comparison on QAP
• Test problems
  • Deceptive ordering problems (relative and absolute).
  • Quadratic assignment problem (QAP) from QAPLIB
  • Assign n facilities to n locations.
  • The user provides flow between pairs of facilities and distances of the locations.
  • We minimize the sum of (flow x distance) for all pairs.

Conclusions
• dtEDA performs better if both relative and absolute ordering constraints are important.
• dtEDA outperforms GA (LOX and PMX) most of the time.
• dtEDA is outperformed by robust tabu search in most random QAP instances (no structure) but not in those with structure.

Acknowledgments
• NSF; NSF CAREER grant ECS-0547013 (at UMSL)
• Ministry of Education, Culture, Sports, Science and Technology of Japan; Grant-in-Aid for Scientific Research No. 19500199
• University of Missouri: High Performance Computing Collaboratory sponsored by Information Technology Services; Research Award; Research Board

Dependency Trees, Permutations, and Quadratic Assignment Problem