DM812
METAHEURISTICS

Lecture 14

Parallel Metaheuristics and Other Topics

Marco Chiarandini

Department of Mathematics and Computer Science
University of Southern Denmark, Odense, Denmark
<marco@imada.sdu.dk>

Outline

1. Parallel Computing
   Introduction
   Parallel Meta-heuristics
2. Optimization under Uncertainty
3. Multi-Objective Optimization
4. Conclusive Notes

Outline

1. Parallel Computing
   Introduction
   Parallel Meta-heuristics
2. Optimization under Uncertainty
3. Multi-Objective Optimization
4. Conclusive Notes

Why Parallelizing?

Goals of parallel implementations:

- reduce running times (speedup: $T_m = T_1/m$)
- robustness
  - partitioning of the search space
  - diversification
  - different strategies and tuning at each processor

Parallel Computing

Architectures:

- Multiple processors, shared memory
  communication through shared memory
- Multiple processors, distributed memories
  communication through network

Logical platform

- SIMD/SISD/MISD/MIMD single/multiple instruction, single/multiple data

Degree of parallelism
granularity: amount of computation between two communication steps

- fine-grained (high speed network or shared memory)
- coarse grained (low speed)

Parallel Computing

Programming Languages

- parallel programming language
  OpenMP, CUDA
  sequence of operations applied to some members of an array
  appropriate for fine-grained
- communication libraries
  PVM, parallel virtual machine; MPI, message passing interface
  one single processor runs one single process suited for coarse-grained computation in clusters
- thread programming
  Java threads
  multiple processors, shared memory
  suited for medium coarse-grained

Parallel Computing

Need for new algorithms: Adaptation of sequential ones may not achieve the best performances.

Parallel algorithms are strongly dependent from the architecture for which they are designed.

Types

- data parallelism
- functional parallelism

- centralized model (master-slave)
- distributed model (all data local to each processor)

Performance Measures

Speedup

$$s_m = \frac{T_1}{T_m} \qquad s_m = \frac{E[T_1]}{E[T_m]}$$

- sublinear speedup ($s_m < m$)
- linear speedup ($s_m = m$)
- superlinear speedup ($s_m > m$)
- Strong speedup: T_1 is the best known sequential algorithm
- Weak speedup: T_1 is taken from the serial version of the parallel algorithm

Note: T can be the time to reach a solution of a certain quality

Performance Measures

Efficiency

$$e_m = \frac{s_m}{m}$$

linear time: $e_m = 1$

$$ie_m = \frac{(m-1)E[T_{m-1}]}{mE[T_m]}$$

A Preliminary Results

Independent multi-thread strategies: Multiple copies of the same sequential algorithm

If the random variable
time to find a solution of a certain quality
is exponentially distributed

$\Downarrow$

linear speedup

Experimentally observed for:

- Simulated annealing, iterated local search, for TSP
- WalkSAT on random 3-SAT instances
- GRASP on max independent set, quadratic assignment, maximum weighted satisfiability, maximum covering []

Parallel Local Search

Use several processors to concurrently explore the neighborhood graph. Load balancing and granularity are in this case problem specific.

- single walk: fine- to medium-grained tasks
  (neighborhood evaluation decomposed and distributed) allows to search larger neighborhoods. Used often with SA
- multiple walks: medium- to coarse-grained tasks
  (multiple trajectories each to a different processor)
  - independent search threads
    (multiple trajectories in the complete neighborhood or problem decomposition by variable fixing)
  - cooperative search threads

Note: the same classification is used for Meta-heuristics

Examples

Single walk parallelization

- TS for a scheduling problem.

  Best improvement $\Rightarrow$ complete neighborhood examination $O(n^2) \times$ function evaluation (makespan) $O(n^2)$

  Neighbors distributed to p processors:

  $$(k-1)\lfloor n/p \rfloor + 1, \ldots, (k-1)\lfloor n/p \rfloor + \lfloor n/p \rfloor \qquad k = 1, \ldots, p-1$$

- Genetic algorithm (if evaluation function is costly)
- On VRP, different clusters of visits to each processor, construction of independent routes.
- No quality improvement expected
- Good load balancing
- Less good with first improvement
- Possible to perform a multiple steps move
  (ie, apply best move found by each processor)

Examples

Multiple walks independent search applied with

- TS to QAP and scheduling
- Ejection chains to VRP
- GRASP to QAP (different random seeds to each processor)
- Scatter Search in island model on QAP
  (but no quality improvement observed)
- Genetic algorithms in island models

Examples

Multiple walks cooperative search.
Shared information gathered in central shared memory or dedicated central processor.
Expected improvements in both quality and speedup.

Which information exchanged?
pool of elite solutions, best solutions and heir costs, attribute frequencies, tabu lists

- Scatter search and Path relinking
- Genetic algorithms
  - island model with communication
  - migration operation policy: processors involved, frequency of exchanges solution exchanged
- Ant Colony
  - several ants to each processor update a central pheromone trail matrix
  - queen process = master
  - hierarchy of several queen processes (island model fashion)
  - overhead

Outline

1. Parallel Computing
   Introduction
   Parallel Meta-heuristics
2. Optimization under Uncertainty
3. Multi-Objective Optimization
4. Conclusive Notes

Optimization Under Uncertainty

In many real life cases problem data might be uncertain.

Approaches (in decreasing order of information available):

- stochastic optimization
- dynamic optimization
- robust optimization
- online optimization

But other factors are relevant to determine the optimization approach.

Stochastic optimization

If probability distributions governing the data are known or can be estimated then it is possible to take advantage of this.

In stochastic optimization the goal is to find some policy that is feasible for all (or almost all) the possible data instances and maximizes the expectation of some function of the decisions and the random variables.

Solution approaches

- The expected value of the objective function can be computed mathematically.

  $$g(s) = E[f(\pi, s)]$$

  where $s \in \mathcal{S}$ and π are the stochastic variables determining the possible scenarios.

  Then no difference with a deterministic problem. Although the evaluation function may become computationally prohibitive.

- Alternatively, the expected value of the objective function can be estimated via sampling and simulation

  $$g(s) = \overline{f}(\pi, s) = \frac{1}{N}\sum_{i=1}^{N} f(\pi_i, s) \qquad \text{(unbiased estimator)}$$

If only uncertainty sets rather than probability distributions are known.

Unlike dynamic and stochastic programming this robust approach does not suffer from the curse of dimensionality

The goal of robust optimization is to find a solution which is feasible for all such data and optimal in some sense.
For example, it optimizes for the worst-case scenario.
Let the uncertain problem be given by

$$\min\{f(x;\pi) : x \in X(\pi)\, \pi \in \Pi\}$$

where Π is some set of scenarios (like parameter values).
The robust optimization model is:

$$\min_x\{\max_{\pi\in\Pi} f(x;\pi) : x \in X(\pi')\, \forall \pi' \in \Pi\}$$

The policy x is required to be feasible no matter what parameter value (scenario) occurs; hence, it is required to be in the intersection of all possible $X(\Pi)$. The inner maximization yields the worst possible objective value among all scenarios.

Outline

1. Parallel Computing
   Introduction
   Parallel Meta-heuristics
2. Optimization under Uncertainty
3. Multi-Objective Optimization
4. Conclusive Notes

Multi-Objective Optimization

A Multi-Objective Combinatorial Optimization Problem (MOCOP) has a vector as objective function $\vec{f} = (f_1, \ldots, f_p)$.

Example

Given an undirected graph $G(V, E)$ with weights $\vec{d}(uv) \in \mathbf{R}^p$ for each edge $uv \in E$.
Find an Hamiltonian cycle H such that

$$\vec{f}(H) = \sum_{uv \in H} \vec{d}(uv)$$

is "minimal".

Elements of a MOCOP

- the set of feasible candidate solutions $\mathcal{S}$
- the objective function vector $\vec{f} = (f_1, \ldots, f_p) : \mathcal{S} \to \mathbf{R}^p$
- the objective space $\mathbf{R}^p$
- the ordered space $(\mathbf{R}^P, \preceq)$
- the model map $\theta : \mathbf{R}^p \to \mathbf{R}^P$

Classification

We distinguish four types of Multiobjective optimization problems according to the requirements:

- min-max optimality ($\theta : \mathbf{R}^p \to \mathbf{R}$)
- weighted sum optimality ($\theta : \mathbf{R}^p \to \mathbf{R}$)
- lexicographic optimality ($\theta : \mathbf{R}^p \to \mathbf{R}$)
- Pareto optimality ($\theta = I : \mathbf{R}^p \to \mathbf{R}^p$)

Min-Max Optimality

$$\min_{s \in \mathcal{S}} \max_{i=1,\ldots,p} f_i(s)$$

Weighted Sum Optimality

$$\min f_w(s) = \sum_{i=1}^{p} w_i f_i(s)$$

Lexicographic optimality

Name	Notation	Definition
lexicographic order	$\vec{y}^1 <_{lex} \vec{y}^2$	$\exists i \in \{1,\dots,p-1\} \mid y^1_k = y^2_k$ $\forall k = 1\dots,i$ and $y^1_{i+1} < y^2_{i+1}$

A solution $s \in \mathcal{S}$ is lexicographic optimal if there is no $s' \in \mathcal{S}$ such that $\vec{f}(s') <_{lex} \vec{f}(s)$.

$$\text{lexmin}_{s\in S}(f_1(s),\dots,f_p(s))$$

$\Rightarrow$ Solve objective sequentially by decreasing order of priority and using the optimal solutions of higher priority objectives as constraints (goal programming).

Pareto Optimality

Definitions on dominance relations

In Pareto sense, for two points (vectors) in $\mathbf{R}^p$

$\vec{y}^1 \preceq \vec{y}^2$	weakly dominates	$y^1_i \leq y^2_i$ for all $i = 1,2,\dots,p$
$\vec{y}^1 \parallel \vec{y}^2$	incomparable	neither $\vec{y}^1 \preceq \vec{y}^2$ nor $\vec{y}^2 \preceq \vec{y}^1$

Hence a set of solutions yields a set of mutually incomparable points (*i.e.*, weakly non-dominated points)

A feasible solution $s \in \mathcal{S}$ is called efficient or Pareto global optimal if there is no other $s' \in \mathcal{S}$ such that $\vec{f}(s') \leq \vec{f}(s)$.

Outline

1. Parallel Computing
   Introduction
   Parallel Meta-heuristics
2. Optimization under Uncertainty
3. Multi-Objective Optimization
4. Conclusive Notes

Craft, Art or Science?

Craft:

- How can heuristic methods be devised for a given problem?
- How can an algorithm be implemented efficiently?
- How can the performance or robustness of the heuristic algorithm be improved?

Answer: Experience and Computer Science Background

Art:

- creativity
- ingenuity
- aesthetic elegance

complement good knowledge and skillful application.
Also similar to software development.

Science

Heuristics are pervasive in many areas: complete search algorithms such as constrain programming, integer programming

- which is the heuristic which works best in a given context?
- why is it so?

The final goal is not practical but intellectual.
Iterated process:

- observe
- formulate hypothesis
- experiment
- build models and theories (evaluated on their explanatory and predictive power and conceptual simplicity). [A model is never identical with what it models, is a heuristic device to enable an understanding of what it models.]

Research Topics

- Engineering heuristic algorithms
  - Advanced data structures
  - Computational studies
  - Simplifications, Organization
- Problem characteristics and search space
  - Phase transition
  - statistical mechanics
- Theoretical results and foundations
  - Run time analysis (worst case, big O)
  - Components utility
  - Neighborhoods connetivity and domination
  - derandomization
- New heuristic methods
  - hybridization with exact methods (CP, IP) [CPAIOR, Matheuristics]
- New applications
  - multiobjective optimization
  - stochastic problems
  - robust optimization