Learning Parallel Portfolios of Algorithms

Marek Petrik

Comenius University

June 7, 2005

Motivation and Principles

- Diverse performance of algorithms on problem instances
- The distribution of instances determines algorithm's performance – unknown during construction
- Processor time is the bottleneck for calculation

Definition (Parallel Portfolio of Algorithms)

- Available algorithms launched in parallel on a single processor
- The share of processor available to each is controlled by a schedule

Goal

Determine the optimal schedules from a training set of instances

Motivation and Principles

- Diverse performance of algorithms on problem instances
- The distribution of instances determines algorithm's performance – unknown during construction
- Processor time is the bottleneck for calculation

Definition (Parallel Portfolio of Algorithms)

- Available algorithms launched in parallel on a single processor
- The share of processor available to each is controlled by a schedule

Goal

Determine the optimal schedules from a training set of instances

Illustration

Example (Traveling Salesman Problem)

- Single optimization problem
- Multiple algorithms, each suitable for different subset of problems
  1. $\mathcal{A}$ Dynamic programming
  2. $\mathcal{B}$ Local search
  3. $\mathcal{C}$ Branch–and–Cut
- Possible schedules
  1. $\mathcal{A}$ 30% $\mathcal{B}$ 50% $\mathcal{C}$ 20% of processor time
  2. $\mathcal{B}$ $\mathcal{B}$ $\mathcal{C}$ $\mathcal{C}$ $\mathcal{A}$ each running for 3 seconds

Formal Definitions

- Problems
  - Optimization – maximize solution quality in fixed time
  - Decision – minimize time, the solution quality is fixed
- Measure of performance on instances
  - Mean Optimization – average performance
  - Limit Optimization – worst case performance
  - Bound Optimization – percentage of instances calculated before a deadline

Schedules

- Static Schedules – Resource allocation constant during the computation
- Dynamic Schedules – Resource allocation changes during the computation in finite discrete intervals

Static Schedules

- Mean and Limit optimization
- Formulation as a mathematical program – hard to solve because it is not continuous

Classification–Maximization Algorithm (CMA)

- Block–coordinate optimization, separation to schedule and classification
- Solvable for specific conditions
- Reaches local minimums, with randomized start

- Optimal CMA – enumerate all classification, high complexity

Dynamic Schedules

- Mean and Bound optimization
- Decision problems only, the execution order does not matter for performance on training

Formulation as a Markov Decision Process

- Calculable by dynamic programming
- Calculation time exponential in the number of algorithms and switches in a schedule

- ϵ–*approximation algorithm* – polynomial in the number of switches and $\frac{1}{\epsilon}$, exponential in number of algorithms

Generalization

- Assure good performance of a PPA on all instances
- Framework motivated by *Probably Approximately Correct* learning
- Distribution free bounds number of samples to achieve $\mathbf{P}\left[\sup_S |P(S) - \mathbf{E}[P(S)]| > \epsilon\right] < \delta$ with polynomial number of samples in $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$

Theorem

The number of samples to learn static and dynamic schedules is polynomial in closeness and certainty of generalization.

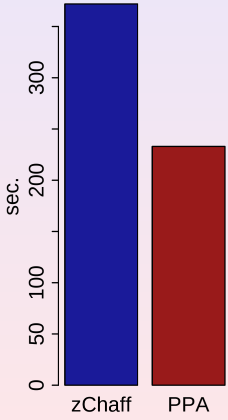
- Bounds polynomial but too wide for practical application

Application: Satisfiability Problem

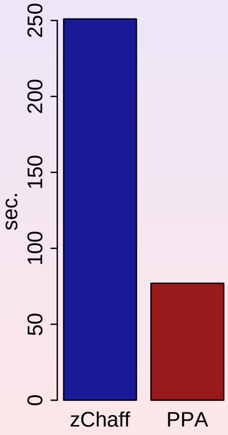
- Satisfiability of a propositional logic formula
- PPA simulated on 1200 instances using 23 algorithms
- The best algorithm – zChaff
- Static Schedules
  - Mean optimization – 3 fold speedup
  - Limit optimization – Solves all instances
- Dynamic Schedules
  - Mean optimization – Narrowly outperformed static
  - Bound optimization – Increased the number of solved instance by 20%
- Generalization results – PPA trained on subsets of instance outperforms zChaff on all

Static Schedule Results

Average run-time on I_1



Average run-time on I_2



Conclusion

- PPA takes advantage of diverse algorithm performance on various instances
- Static schedules are simple to calculate using CMA
- Dynamic schedules can be calculated for a small number of algorithm
- Application on SAT indicates PPA may significantly increase the performance
- Good theoretical and practical generalization properties

General Mean Optimization Problem

$$
\begin{aligned}
\text{maximize} \quad P(S) &= \sum_{i=1}^{m} \max_{j=1,\ldots,n} p_j(r_j, x_i) \\
\text{subject to} \quad \sum_{j=1}^{n} r_j &= 1, \\
r_j &\geq 0 \quad j = 1, \ldots, n
\end{aligned} \tag{1}
$$

- Inner max operator makes the objective function discontinuous
- Unsolvable by standard optimization methods

Possible Reformulation

$$
\begin{aligned}
\text{maximize} \quad P(S, W) &= \sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij} p_j(r_j, x_i) \\
\text{subject to} \quad \sum_{j=1}^{n} r_j &= 1, \\
\sum_{j=1}^{n} W_{ij} &= 1 \quad i = 1, \ldots, m, \\
r_j &\geq 0 \quad j = 1, \ldots, n, \\
W_{ij} &\in \{0, 1\} \quad i = 1, \ldots, m \quad j = 1, \ldots, n
\end{aligned} \tag{2}
$$

CMA Approach for Mean Optimization

Classification Phase

$$\begin{aligned} \text{maximize} \quad P(W) &= \sum_{i=1}^{m}\sum_{j=1}^{n} W_{ij}\, p_j(r_j, x_i) \\ \text{subject to} \quad \sum_{j=1}^{n} W_{ij} &= 1 \quad i = 1, \ldots, m, \\ r_j &\geq 0 \quad j = 1, \ldots, n. \end{aligned} \tag{3}$$

CMA Approach for Mean Optimization

Maximization Phase

$$
\begin{aligned}
\text{maximize} \quad P(S) &= \frac{1}{m}\sum_{j=1}^{n} \nu_j(r_j) d_j \\
\text{subject to} \quad \sum_{j=1}^{n} r_j &= 1 \\
r_j &\geq 0 \quad j = 1, \ldots, n
\end{aligned} \tag{4}
$$