Introduction to algorithm portfolio scheduling

Yanik Ngoko

IME-USP

2011
1 Introduction
   - Context
   - Objectives

2 The discrete Resource Sharing Scheduling Problem
   - The problem
   - Analysis

3 Solving the dRSSP
   - Complete exploration
   - Heuristics for dRSSP
   - Mean allocation based heuristics
Poly-algorithms

The evolution of algorithmic is leading to a huge number of algorithms available for the resolution of a same computational problems. There is a need to converge towards poly-algorithms that automatically combine a set of elementary algorithms in a superior ones.

Some main models

- Algorithm selection models
- Algorithm cascading
- Algorithm ranking
- Algorithm portfolio.
We revisit in this course the main computational learning problems for combining algorithms under algorithm portfolio. These are: the discrete Resource Sharing Scheduling Problem, the Time Sharing Scheduling Problem and the Malleable sharing scheduling problem.
The decision variant can be defined as follows. 

**Instance**: A finite set of instances \(\mathcal{I} = \{I_1, \ldots, I_n\}\), A finite set of algorithms \(\mathcal{A} = \{A_1, \ldots, A_k\}\), a set of \(m\) identical resources, Cost values \(C(A_i, I_j, p) \in R^+\) for each \(I_j \in \mathcal{I}, A_i \in \mathcal{A}\) et \(p \in \{1, \ldots, m\}\), a positive real \(T\).

**Question**: Is there a vector \(S = (S_1, \ldots, S_k)\) where \(S_i \in \{0, \ldots, m\}\) and \(0 < \sum_{i=1}^k S_i \leq m\) such that \(R_{\text{sum}}\{C, S\} = \sum_{j=1}^n \min_{1 \leq i \leq k} \{C(A_i, I_j, S_i)|S_i > 0\} \leq T\)?

Another objective is \(R_{\text{max}}\{C, S\} = \max_{A_i, I_j} \{C(A_i, I_j, S_i)|S_i > 0\}\).
EXAMPLES

Example 1
Let us assume that we have $|I| = m = |A|$. $\forall A_i, l_j, p$

\[
C(A_i, l_j, p) = \begin{cases} 
1 & \text{if } i = j \\
2m & \text{otherwise} 
\end{cases}
\]

what is the vector minimizing $R_{\text{sum}}$ and $R_{\text{max}}$?

Example 2
$C(A_i, l_j, m)(m = 3)$ is given by: $C = \begin{pmatrix} 14 & 1 & 14 \\ 1 & 2 & 10 \\ 14 & 14 & 2 \end{pmatrix}$. Each line $j$ here gives the costs $C(A_1, l_j, m), C(A_2, l_j, m), C(A_3, l_j, m)$. $C(A_i, l_j, p) = \frac{m}{p} C(A_i, l_j, m)$. what is the vector minimizing $R_{\text{sum}}$ and $R_{\text{max}}$?
**Geometric Representation**

- **Res.**
  - $I_1$
  - $I_2$
  - $I_3$

- **Time**
  - $t_1$
  - $t_2$
  - $t_3$

- **Resources**
  - $S_1$
  - $A_1$
  - $S_2$
  - $A_2$
  - $S_3$
  - $A_3$
The resolution of any computational instance under dRSSP requires an amount of work that we can classify in two groups: useful work leading to the resolution of the instance and useless one.

For minimizing useless work, we proceed by decomposing the dRSSP in two sub-problems:

- An assignment problem that states for each instance the best algorithm that will on it provide the first solution.
- A load balancing problem that given an assignment (instances-algorithm) find the best allocation of resource.
**FORMALIZATION OF THE ASSIGNMENT PROBLEM**

**Definition**

We define a workload distribution by an injection \( \sigma : \mathcal{D} \) where \( \mathcal{D} \equiv \mathcal{I} \rightarrow \mathcal{A} \).

There can be an algorithm for which there is no assigned instance. Given a distribution \( \sigma \) and an instance \( I_j \), \( \sigma(I_j) \) gives the algorithm that has a useful execution on \( I_j \).

**Definition**

Given a workload distribution \( \sigma \), we define the workload denoted by \( A_i \) as \( W(\sigma, A_i) = \sum_{I_j \in \sigma^{-1}(A_i)} C(A_i, I_j) \). It captures the sum of sequential execution times that \( A_i \) can take for solving instances assigned to it from the \( \sigma \) distribution.
We formulate the problem of finding the optimal workload distribution through the Workload Estimation Problem (WEP).

\[
\text{Minimize } \left( \sum_{j=1}^{n} W(\sigma, l_j), |A^+| \right)^t
\]

1. \( \sigma \in D \)
2. \( A^+ = \{ A_i \in A | \sigma^{-1}(A_i) \neq \emptyset \} \)
Introduction to algorithm portfolio scheduling
The discrete Resource Sharing Scheduling Problem
Solving the dRSSP

COMPLEXITY RESULTS

**Theorem**

If we fix the second criteria, then WEP is NP complete and inapproximable

The reduction to the set cover problem. It is important to remark that the result holds even if $|A^+| = |A|$.

**Corollary**

the dRSSP is NP complete and inapproximable within a constant factor.

Reduction to WEP.
Formalizing the Load balancing problem

**Instance** : A finite set of $k$ instances subsets $= \{ G_1, \ldots, G_k \}$, a finite set of algorithms $A = \{ A_1, \ldots, A_k \}$, a set of $m$ identical resources, a cost $C(A_i, l_j, p) \in R^+$ for each $l_j \in G_i$, $A_i \in A$ and $p \in \{1, \ldots, m\}$, a real value $T \in R^+$.

**Question** : Is there a vector $S = (S_1, \ldots, S_k)$ with $S_i \in \{0, \ldots, m\}$ and $0 < \sum_{i=1}^{k} S_i \leq m$ such that $\sum_{i=1}^{k} \sum_{l_j \in G_i} \{ C(A_i, l_j, S_i) \} \leq T$?
The Load balancing problem can be solved by dynamic programming. On the $R_{min}$ case, we consider a dynamic programming table that progressively computes values of a matrix $Cost(\cdot, \cdot)$.

Each value $Cost(u, m_u)$ is the minimal cost for solving the instances $\bigcup_{1 \leq i \leq u} G_i$ in using a total of $m_u$ resources.

This means that each $Cost(u, m_u)$ must correspond to a partial assignment of $m_u$ resources to the algorithms $A_1, \ldots, A_u$. 
SOLVING THE LOAD BALANCING PROBLEM

We use the following Bellman equations:

\[
Cost(u, m_u) = \sum_{i=1}^{u} \sum_{l_j \in G_i} \{C(A_i, l_j, S_i)\}. \text{ and}
\]

\[
Cost(u, m_u) = \begin{cases} 
\min_{0 \leq m_{u-1} \leq m_u} \{Cost(u - 1, m_{u-1}) + \\
\sum_{l_j \in G_u} C(A_u, l_j, m_u - m_{u-1})\} & \text{if } u > 1 \\
\sum_{l_j \in G_1} C(A_1, l_j, m_1) & \text{if } u = 1.
\end{cases}
\]

**Exercice**

What are the Bellman equations in the case of \( R_{max} \) minimization?
## Solution sets

1. A solution to dRSSP is a vector $S \in \{0, \ldots, m\}^k$
2. There is at most $(m + 1)^k$ solutions

If $k$ is small, we can by a complete exploration find a solution in a reasonnable amount of time.
Exploration Principle

- At each time during the exploration, we have a vector $S$ and an index $u$ such that the $u - 1$ first components $(S_1, \ldots, S_{u-1})$ comprise an assignment of $\text{sum} \geq m$ resources. We want to assign an element to $S_u$.
- We pick a number of resource between 0, $m - \text{sum}$, assign it to $S_u$ and increase the value of $u$.
- If we have assigned all resources, we compute the resulting $R_{min}$ and compare it to the best previous computed values. Otherwise, we increment $u$.

The principle is recursive.
EXERCICES

EXERCICE

From the above principle, derive a complete exploration algorithm for dRSSP. Propose a recursive version and then an iterative one.

Analyze its time complexity.

EXERCICE

Propose a version of the above algorithms specialized on the case where all $A_i$ are sequential.
There are two classes of heuristics for dRSSP:

- Mean allocation or proportional allocation based heuristics.
- Heuristics based on (assignment-load balancing)

The principle of mean allocation is to equally (or near) share the resources among algorithms. We will assume on this class of algorithm that we have a linear speed-up i.e.

\[ C(A_i, I_j, p) = \frac{C(A_i, I_j, m)m}{p} \]
Mean Allocation (MA)

We randomly choose a subset $\mathcal{A}^* \subseteq \mathcal{A}$ of algorithms that can have the shortest execution time on at least one resource. Let us assume that this set comprises the algorithms $A_1, \ldots, A_u$. We have the following algorithm.

**MA($S$)**

1. $q = m \div k$
2. $r = m - k \times q$
3. For $i = 1$ to $u$
   - $S_i = q$
4. For $i = 1$ to $r$
   - $S_i = S_i + 1$
Mean Allocation analysis

Let $p_{\text{max}}$ be the maximal number of resources that can be allocated to a single algorithm.

**Lemma**

MA is a $(2u - 1). \frac{p_{\text{max}}}{m}$ approximation for dRSSP in $O(k)$

Idea in the proof: compare with the lower bound

$Lb = \frac{m}{p_{\text{max}}} \sum_{j=1}^{n} \min_{i} \{C(A_{i}, l_{j}, m)\}$.

The shortest is $A^*$, the better is the bound. This results also states that in all cases, we have in the worst case a $2k - 1$ to dRSSP.
Mean Allocation with Guess (biMAG)

It is a bi-dimensional exploration based on two parameters \((g, d)\) \((g \in \{1, \ldots, k\}, d \in \{1, \ldots, m\})\). \(g\) serves to reduce the exploration on the number of algorithms and \(d\) reduces it on the number of resources. The exploration is done as follows: initially, one chooses randomly a subset \(A_g \in \mathcal{A}\) of \(g\) algorithms. If \((d \geq 2)\), one considers all resources sharing assigning to heuristics in \(A_g\), a number of resources in the set \(\{0, d^0, \ldots, d^f\}\) where \(d^f \leq m < d^{f+1}\). For any allocation of \(m_g\) resources to algorithms in \(H_g\), one only considers the allocations for \(\mathcal{A} \setminus A_g\) where a subset \(A'_{k-g} \subseteq (\mathcal{A} \setminus A_g)\) has \(\lfloor \frac{m}{A'_{k-g}} \rfloor\) resources on the remaining \(\mathcal{A} \setminus A'_{k-g}\) algorithms no resource. In the case where \(d = 1\), we proceed as previously with the only difference that for each algorithms in \(A_g\), we will consider the allocations among the set \(\{0, 1, \ldots, m\}\).
**Theorem**

The bi-dimensional exploration with parameter \((g, d)\) provides a solution of quality \(k - g + d\) in time \(O(n 2^{k-g} (\log_d(m + 1))^g)\).

**Exercice**

Analyze the approximation ratio on \(R_{max}\).