Electronic Colloquium on Computational Complexity, Report No. 82 (2006)

Polynomial-Time Maximisation Classes: Syntactic Hierarchy

P. Manyem
Centre for Informatics and Applied Optimisation,
School of IT and Mathematical Sciences,
University of Ballarat,
Mount Helen, VIC 3350, Australia.
mailto: pmanyem@staff.ballarat.edu.au

Abstract

In Descriptive Complexity, there is a vast amount of literature on decision problems, and their classes such as **P, NP, L and NL**. However, research on the descriptive complexity of optimisation problems has been limited. In a previous paper [Man], we characterised the optimisation versions of **P** via expressions in second order logic, using universal Horn formulae with successor relations. In this paper, we study the syntactic hierarchy within the class of polynomially bound maximisation problems. We extend the result in the previous paper by showing that the class of polynomially-bound **NP** (not just **P**) maximisation problems can be expressed in second-order logic using Horn formulae with successor relations. Finally, we provide an application — we show that the Bin Packing problem with online LIB constraints can be approximated to within a $\Theta(\log n)$ bound, by providing a syntactic characterisation for this problem.

1 Introduction

Descriptive Complexity (DC) began with Fagin's 1974 theorem [Fag74], which captures the class **NP** as the set of properties that can be represented in existential second order logic. Approximation complexity measures how well an NP-hard optimisation problem can be approximated, or how far is the value of a (possible) heuristic solution from that of an optimal solution. This paper attempts to connect the two complexity measures.

A few previous attempts to characterise approximation classes in terms of logic (DC) are: Papadimitriou and Yannakakis in 1991 [PY91], Panconesi and Ranjan in 1993 [PR93], Kolaitis and Thakur in 1994 and 1995 [KT94, KT95], Khanna et al in 1998 [KMSV98], and Manyem [Man].

In [KT94, PR93, PY91], the authors characterise approximation hardness in terms of quantifier complexity — the number and types of quantifiers that appear at the beginning of a second-order formula in prenex normal form (PNF). For a formula in PNF, all quantifiers appear at the beginning, followed by a quantifier-free formula.

1.1 Contributions

In a previous paper [Man], we presented
(a) a logical representation of a subclass of $\mathbf{P}'$ — $\mathbf{P}'$ is the class of optimisation problems

ISSN 1433-8092

σ	vocabulary
$\mathbf{A}$	a structure defined over σ (captures an instance of an optimisation problem)
η	a quantifier-free first order formula, and a conjunction of *Horn* clauses at the same time. (Recall that a Horn clause contains at most one positive literal.)
$\mathbf{x}$	an $m-$tuple of first order variables
$\mathbf{S}$	a sequence of second-order variables (predicate symbols) (captures a solution to the optimisation problem)
$\mathbf{P}$	computational class of *decision* problems, decidable in polynomial time by a deterministic Turing machine
$\mathbf{P'}$	class of *optimisation* problems corresponding to $\mathbf{P}$ (also called $\mathbf{P}-$optimisation problems)
$\mathbf{Q'}$	$\mathbf{Q'} \subseteq \mathbf{P'}$, and $\mathbf{Q'}$ only contains *polynomially bound* optimisation problems (see Definition 1)
$\mathbf{N}$	class of *optimisation* problems whose decision versions are in $\mathbf{NP}$
$\mathbf{N'}$	$\mathbf{N'} \subseteq \mathbf{N}$, and $\mathbf{N'}$ only contains *polynomially bound* optimisation problems
ESO	Existential Second Order Logic
PNF	Prenex Normal Form

Table 1: Notation

that can be solved to optimality within polynomial time[1]. The class of *decision problems* corresponding to $\mathbf{P'}$ is $\mathbf{P}$. The syntactic characterisation of $\mathbf{P}$ is given below in Theorem 1, due to Grädel [E. 91].
(b) The particular subclass $\mathbf{Q'}$ (of $\mathbf{P'}$) that we focus on includes only *polynomially bound optimisation problems*, defined below in Definition 1. We provided syntactic characterisations for both maximisation and minimisation problems in $\mathbf{Q'}$.
(c) gave examples of characterisations (MAXFLOW$_{PB}$ for maximisation and SHORTEST PATH$_{PB}$ minimisation),
(d) showed that MAXFLOW$_{PB}$ is complete for the maximisation subclass of $\mathbf{Q'}$,
(e) presented characterisations for maximisation problems in $\mathbf{P'}$ (defined in Table 1 — problems not necessarily polynomially bound), as well as an example for a problem in this class (MAXIMUM MATCHING). This is a considerable departure from the treatment in [Zim98].

1.2 Notation and Definitions

All notation is defined in Table 1, as a one-stop reference point. For the same reason, all definitions are provided below in this section.

[1]Strictly speaking, in Turing machine terminology, $\mathbf{P'}$ is the set of languages where, if an instance I of an optimisation problem $P \in \mathbf{P'}$ is encoded as an input string x in some alphabet Σ, a deterministic Turing machine will compute the optimal solution (which is again a string) within $\Theta(|x|^k)$ steps, where k is some constant and $|x|$ is the length of the input string.

Definition 1. *An optimisation problem Q' is said to be* ***polynomially bound*** *if the value of an optimal solution to every instance I of Q' is bound by a polynomial in the size of I. In other words, there exists a polynomial p such that*

$$opt_{Q'}(I) \leq p(|I|), \tag{1}$$

for every instance I of Q'. The class of all such problems is $\mathbf{Q'}$.

Definition 2. ***First order logic*** *consists of a vocabulary (alias signature) σ, and models (alias structures) defined on the vocabulary. In its simplest form, a vocabulary consists of a set of variables, and a set of relation symbols $R_j (1 \leq j \leq J)$, each of arity r_j. A model M consists of a universe U whose elements are the values that variables can take — M also instantiates each relation symbol $R_j \in \sigma$ with tuples from $U^{(r_j)}$. For example, a model* $\mathbf{G}$ *in graph theory may have the set of vertices $G = \{1, 2, \cdots 10\}$ as its universe (assuming that the graph has 10 vertices), and a single binary relation E where $E(i, j)$ is true iff (i, j) is an edge in the graph* $\mathbf{G}$. *A model represents an instance of an optimisation problem.*

Definition 3. *A* $\mathbf{\Pi_1}$ $(\mathbf{\Sigma_1})$ ***first order formula in PNF*** *only has universal (existential) quantifiers, quantified over first order variables.*

Definition 4. *A formula in* ***existential second-order (ESO) Logic*** *is of the form $\phi \equiv \exists \mathbf{S} \psi$, where ψ is a first order formula, and* $\mathbf{S}$ *is a sequence $\{S_1, S_2, \cdots, S_p\}$ of relation symbols not in the vocabulary of ψ. The formula ϕ can be written as*

$$\phi = \exists \mathbf{S} \psi = \exists S_1 \cdots \exists S_p \psi. \tag{2}$$

In an ***ESO Horn*** *expression $\exists \mathbf{S} \psi$, the first order formula ψ can be written in Π_1 form as*

$$\psi = \forall x_1 \forall x_2 \cdots \forall x_k \eta = \forall \mathbf{x}\ \eta. \tag{3}$$

where η is a conjunction of Horn clauses (η is, of course, quantifier-free), and x_i $(1 \leq i \leq k)$ are first order variables. Each clause in η contains at most one positive occurrence of any of the second order predicates S_i $(1 \leq i \leq p)$.

Definition 5. *A* $\mathbf{\Pi_2}$ $(\mathbf{\Sigma_2})$ ***formula in prenex normal form (PNF)*** *can be written as*

$$\phi = \forall x_1 \cdots \forall x_a\ \exists y_1 \cdots \exists y_b\ \eta \quad (\phi = \exists y_1 \cdots \exists y_b\ \forall x_1 \cdots \forall x_a\ \eta), \tag{4}$$

where η is quantifier-free, $a, b \geq 1$, and the x's and y's are first-order variables.

The following theorem is due to Grädel [E. 91] — this is the polynomial-time counterpart of Fagin's theorem [Fag74] which characterised the class **NP**:

Theorem 1. *For any ESO Horn expression as defined in Definition 3, the corresponding decision problem is in* $\mathbf{P}$.

The converse is also true — if a problem P is in $\mathbf{P}$, *then it can be expressed in ESO Horn form — but only if a successor relation is allowed to be included in the vocabulary of the first-order formula ψ.*

1.3 Background

An example of a syntactic characterisation is the number of *alternations* of quantifiers at the front of a logical expression — the complexity class that a problem belongs to, depends on how many times quantifiers alternate in expressions.

Examples:

1. In the expression for ϕ_1, where $\phi_1 : \exists x_1 \exists x_2 \exists x_3 \psi$, if ϕ_1 is in PNF, then it has three existential quantifiers $\exists x_1$, $\exists x_2$, and $\exists x_3$. Since there is only one type of quantifier (existential, or $\exists$), this expression has no *quantifier alternation*. The ψ term is the *quantifier-free* part of ϕ_1.

   (Such a formula is said to be in Σ_1 form. The Σ signifies that the sequence of quantifiers begins with a set of existentials at the left end.)

2. In ϕ_2, where $\phi_2 : \exists x_1 \exists x_2 \exists x_3 \forall x_4 \forall x_5 \psi$, we have one *quantifier alternation*, when we move from $\exists x_3$ (existential) to $\forall x_4$ (universal). There are two universal quantifiers, $\forall x_4$ and $\forall x_5$. As before, ψ is the quantifier-free part of ϕ_2.

   (Such a formula is said to be in Π_2 form. The Π signifies that the sequence of quantifiers begins with a set of universals at the left end. The **2** in Π_2 signifies that there are two uniform sub-sequences of quantifiers, $\exists x_1 \exists x_2 \exists x_3$ and $\forall x_4 \forall x_5$.)

3. In general, a Σ_n (Π_n) formula in PNF form begins with a subsequence of existential (universal) quantifiers, has $n-1$ alternations of quantifiers, followed by a quantifier-free formula.

2 Polynomially Bound P-Maximisation Problems in Q′

Optimisation problems corresponding to P. We assume that for a maximisation (or a minimisation) problem Q' in the class $\mathbf{Q'}$ (corresponding to the class $\mathbf{P}$ of decision problems), the following can be computed in polynomial time deterministically: (a) The value of the objective function $f(\mathbf{A}, \mathbf{S})$ to a solution $\mathbf{S}$ of an instance $\mathbf{A}$, and (b) Whether a solution $\mathbf{S}$ is a feasible solution to an instance $\mathbf{A}$.

For maximisation problems in $\mathbf{N'}$ (see Table 1 for a definition of $\mathbf{N'}$), Kolaitis and Thakur [KT94] proved the following:

Theorem 2. *A maximisation problem $Q \in \mathbf{N'}$ if and only if there exists a Π_2 first order formula $\phi(\mathbf{w}, \mathbf{S})$ with predicate symbols from the vocabulary σ (of ϕ) and the sequence $\mathbf{S}$, such that for every instance $\mathbf{A}$ of Q, the optimal solution value is given by*

$$opt_Q(\mathbf{A}) = \max_{\mathbf{S}} | \{\mathbf{w} : (\mathbf{A}, \mathbf{S}) \models \phi(\mathbf{w}, \mathbf{S})\} |. \tag{5}$$

In other words, polynomially bound NP-maximisation problems fall in what is called the MAX Π_2 class. In [Man], the author showed a similar result for the polynomial-time counterpart of $\mathbf{N'}$, that is, maximisation problems in $\mathbf{Q'}$:

Theorem 3. *Let* $\mathbf{A}$ *be a structure (instance) defined over* σ*. The value of an optimal solution to an instance* $\mathbf{A}$ *of a maximisation problem* Q' *can be represented by*

$$opt_{Q'}(\mathbf{A}) = \max_{\mathbf{S}} |\{\mathbf{w} : (\mathbf{A}, \mathbf{S}) \models \forall \mathbf{x}\ \eta(\mathbf{w}, \mathbf{x}, \mathbf{S})\}| \tag{6}$$

if $Q' \in \mathbf{Q}'$*, where* $\mathbf{x}$*,* $\mathbf{A}$*,* $\mathbf{S}$ *and* η *are defined in Table 1.*

The proof of Theorem 3 uses Grädel's theorem (Theorem 1) as a starting point.

Notes. Theorem 3 does NOT provide a polynomial-time algorithm to obtain an optimal solution — the maximum in (6) is taken over ALL solutions $\mathbf{S}$, which could be exponential in number. Though the number of *solution values* is at most $|A|^{\mathcal{R}}$, which is polynomial in the size of the instance, the number of *solutions* need not be. Examining each solution value between 1 and $|A|^{\mathcal{R}}$ does not guarantee finding a solution in polynomial time[2].

2.1 Applying Grädel's Theorem to NP-Maximisation Problems

It is possible to express the optimal values for polynomially bound NP-maximisation problems using ESO Horn expressions where the first order vocabulary includes a successor relation (which imposes a linear order on the universe). The proof presented here is due to Radhakrishnan [Rad05].

Let Q be an NP-maximisation problem. The maximum solution value for an instance I to Q is given by

$$g_Q(I) = \max_T f_Q(I, T), \tag{7}$$

where we maximise over all solutions T to I. We assume the following here:

- If T is not a feasible solution to I, then $f_Q(I, T) = 0$.
- $f_Q(I, T) \geq 0$ for all feasible I.
- $f_Q(I, T)$ is a function computable in time polynomial in $\|I\| + \|T\|$ (the size of I + the size of T).

Given an NP-maximisation problem Q, an instance I of Q, and a relation W defined on the universe A of I, a decision problem R_Q corresponding to Q would be the following:

$$\begin{aligned} R_Q(I, W): \quad & \text{Does } Q \text{ possess a feasible solution } T \text{ such that} \\ & f_Q(I, T) \geq |W| \text{ (the cardinality of } W) \text{ ?} \end{aligned} \tag{8}$$

Clearly, $R_Q(I, W) \in \mathbf{NP}$. However, if solution T is given, then the following decision problem $\mathcal{R}_Q \in \mathbf{P}$,

$$\mathcal{R}_Q(I, T, W): \text{ Is } f_Q(I, T) \geq |W| \text{ ?} \tag{9}$$

since the Turing Machine deciding R_Q no longer needs to guess T — it only needs to check the condition (as to whether $f_Q(I, T) \geq |W|$) and return a yes/no answer, which can be done deterministically in polynomial time, since $R_Q \in \mathbf{NP}$.

[2]The difficulty arises in "inverse mapping" solution values to solutions.

According to Grädel, the decision problem $\mathcal{R}_Q(I,T,W)$ can be represented in ESO Logic, where the first order part is represented by a conjunction of Horn clauses and a successor relation is included in the first-order vocabulary. That is, if a given instance (I,T,W) is a *yes* instance to $\mathcal{R}_Q(I,T,W)$, then

$$(I,T,W) \models \exists \mathbf{S}\ \forall \mathbf{x}\ \eta(\mathbf{x},\mathbf{S},I,T,W). \tag{10}$$

Hence, applying Theorem 3 to (10), we can say that

Theorem 4. *The maximum value for the NP-maximisation problem Q can be expressed as*

$$g_Q(I) = \max_T f_Q(I,T) = \max_{\mathbf{S},T,W} |\{\mathbf{w} : (I,T,W,\mathbf{S}) \models \forall \mathbf{x}\ \eta(\mathbf{x},\mathbf{S},I,T,W) \wedge W(\mathbf{w})\}|. \tag{11}$$

A similar result can also be proven for NP-minimisation problems.

2.2 Example: Polynomially Bound Maximum Flow (Unit Capacities)

In [Man], we showed the following three properties:

(a) $\mathbf{MAXFLOW_{PB}}$. The MAXFLOW problem with unit capacities can be expressed in ESO, in Π_1 form. Given a source s and a sink t, and a network G containing directed edges, we want to find the maximum flow that can be sent through the network from s to t. Essentially we seek the maximum number of edge-disjoint paths from s to t. This problem is polynomially bound.

In most such expressions, there are two types of conditions to be expressed: (1) *Global* conditions (those that apply over all $\mathbf{w}$ tuples), and (2) *Local* conditions (the ones that are specific to a given $\mathbf{w}$). The first (second) set of conditions correspond to *constraints* (*objective function*) in a classical mathematical programming framework.

(b) The MAXFLOW$_{PB}$ problem is complete for the class of polynomially bound maximisation problems — an instance I of a general problem Q' in this class can be reduced (in polynomial time) to an instance $\mathcal{I}$ of MAXFLOW$_{PB}$.

(c) The property, that an edge belongs at most one edge-disjoint $s-t$ path in a solution, (and hence the MAXFLOW$_{PB}$ problem) can be expressed with a Π_1 formula, but not with a Σ_1 formula.

2.3 A Problem in $\mathbf{MAX_P\Sigma_0}$?

Within the class $\mathbf{Q'}$ (see Table 1), let us define the class MAX$_P\Sigma_0$ (MAX$_P\Pi_1$) as the class of polynomially bound maximisation problems that can be expressed by a Σ_0 (Π_1) formula.

Kolaitis and Thakur showed that MAX3SAT is in MAX$_{NP}\Sigma_0$ (defined similar to MAX$_P\Sigma_0$), a subset of $\mathbf{N'}$. On the other hand, MAX2SAT is not known to be polynomially solvable [H$\mathring{\text{o}}$97], though the decision version, as to whether all clauses are satisfiable, is well-known to be in $\mathbf{P}$ [GJ79]. Results similar to MAX2SAT for both the maximisation and decision versions are also known for HORNSAT (where every clause is required to be a Horn clause) [KKM94].

Towards the goal of obtaining a hierarchy within the polynomially bound P-maximisation class, we need to exhibit a problem in $\mathrm{MAX}_P\Sigma_0$. However, we have been unable to find a "genuine" maximisation problem in this category so far, except for "trivial" optimisation problems such as MAX1SAT (MAX-one-SAT) defined below.

Definition 6. ***MAX1SAT***.
Given. A set of clauses c_i, $1 \leq i \leq n$. Each clause c_i is either a Boolean variable x_i or its negation $\neg x_i$. (Either $c_i = x_i$ or $c_i = \neg x_i$.)
Problem. Assign truth values to the x_i's such that the number of satisfied clauses is maximised.

Syntactic Characterisation. An instance $\mathbf{A}$ of the problem consists of a universe $\mathrm{A} = \{x_1, x_2, \cdots, x_n\}$ (the n variables, corresponding to the n clauses). A first-order unary relation P is defined on each x_i — $P(x_i)$ is true if clause $c_i = x_i$ and false if $c_i = \neg x_i$. A second-order unary relation S is defined such that $S(x_i)$ holds iff variable x_i is assigned to be true. The objective is to maximise the number of clauses w such that

$$opt_{MAX1SAT}(\mathbf{A}) = \max_S |\{w : (\mathbf{A}, S) \models \ P(w) \Leftrightarrow S(w)\}|\,, \tag{12}$$

the obvious optimal solution value being n.

We conjecture that as long as a successor relationship or a linear ordering on the universe of a structure is necessary, a problem cannot be expressed in $\mathrm{MAX}_P\Sigma_0$ (since this will require a Π_1 expression). We next show that the LINEAR EQUALITIES (LE) problem is in $\mathrm{MAX}_P\Sigma_1$.

2.4 Linear Equalities or LE (a problem in $\mathbf{MAX_P\Sigma_1}$)

Given. A set of linear equalities of the form

$$\sum_{j=1}^{n} a_{i,j}x_j = c_i, \quad 1 \leq i \leq m, \text{ or simply } Bx = c, \tag{13}$$

a finite structure $\mathbf{A}$, with a finite contiguous subset of the natural numbers ($\mathbf{N} = \{0, 1, \cdots\}$) as the universe, and with relations $PLUS(a, b, c)$ (which holds iff $a + b = c$), and $MULT(a, b, c)$ (which holds iff $a \times b = c$). The matrix B is of size $m \times n$.

Problem. Determine the maximum number of solutions to the given set of linear equalities.

If
(i) $m = n = 2$,
(ii) the determinant of B is not equal to zero; that is, $a_{1,1}a_{2,2} - a_{1,2}a_{2,1} \neq 0$, and
(iii) $c_1 \neq 0$ and $c_2 \neq 0$,
then the system has a unique solution. Let this simpler system consist of the two equalities $a_1x_1 + b_1x_2 = c_1$ and $a_2x_1 + b_2x_2 = c_2$ (if we define the simpler notation $a_{1,1} = a_1$, $a_{1,2} = b_1$, $a_{2,1} = a_2$, and $a_{2,2} = b_2$).

The optimal solution value (which equals one) can be expressed as

$$opt_{LE}(\mathbf{A}) = \max_S |\{(x_1, x_2) : (\mathbf{A}, \mathbf{S}) \models \exists y_1 \cdots y_4 z_1 z_2 \ \Phi\}|, \tag{14}$$

where

$$\begin{aligned}\Phi = \quad & MULT(a_1, x_1, y_1) \wedge MULT(b_1, x_2, y_2) \wedge \\ & MULT(a_2, x_1, y_3) \wedge MULT(b_2, x_2, y_4) \wedge \\ & PLUS(y_1, y_2, c_1) \wedge PLUS(y_3, y_4, c_2) \wedge \\ & [c_1 \neq 0] \wedge [c_2 \neq 0] \wedge \\ & MULT(a_1, b_2, z_1) \wedge MULT(a_2, b_1, z_2) \wedge [z_1 \neq z_2].\end{aligned} \tag{15}$$

The above expression for the optimal value can easily be extended to general (but finite) values of m and n. Even if conditions (ii) and (iii) do not hold, and the number of solutions to the linear system is more than one and finite, we can still obtain a Σ_1 expression for Φ.

Clearly, LINEAR EQUALITIES cannot be expressed in Σ_0 form — the PLUS and MULT relations need an existential quantifier, without which the expression Φ would no longer be independent of the size of an instance. Hence, $\text{MAX}_P\Sigma_0 \subset \text{MAX}_P\Sigma_1$.

2.5 Hierarchy Within Maximisation

From Section 2.2, we know that MAXFLOW_{PB} cannot be expressed in Horn Σ_1 form, hence $\text{MAX}_P\Sigma_1 \subset \text{MAX}_P\Pi_1$. The fact that $\text{MAX}_P\Sigma_1 \subseteq \text{MAX}_P\Pi_1$ is clear from Theorem 3. We state a strict hierarchy within the polynomially bound maximisation class (since it is clear from the arguments above):

Lemma 5. *The $MAX_P\Sigma_0$ ($MAX_P\Sigma_1$) class is strictly contained within the $MAX_P\Sigma_1$ ($MAX_P\Pi_1$) class. That is, $MAX_P\Pi_0 \subset MAX_P\Sigma_1 \subset MAX_P\Pi_1$.*

From Section 2.2, we know that MAXFLOW_{PB} serves as a complete problem for the $\text{MAX}_P\Pi_1$ class. It would be desirable to obtain a complete problem for the $\text{MAX}_P\Sigma_0$ and $\text{MAX}_P\Sigma_1$ classes. An interesting observation is that the decision version of the weighted MAXFLOW problem (where arc capacity can be any non-negative integer) is complete for the class **P** [GHR95, Imm99].

3 Bounds Using Descriptive Complexity

In this section, we show that the OLIBP (online uniform sized bin packing with LIB constraints) can be approximated to within a $\Theta(\log n)$ bound, where n is the input size, using techniques from Descriptive Complexity (DC). Item sizes are discrete (for example, in steps of 0.05). Depending on the syntactic characterisation, it is possible to determine how hard it is to approximate a problem.

Problem Statement: Online LIB Variable-Sized Bin Packing (OLIBP). Given an infinite supply of variable sized bins, and a list of n items. The i^{th} item has a size a_i from the set of sizes $\{t_1, \cdots, t_k\}$, where $0 < t_1 < t_2 < \cdots < t_k = 1$. Each item should be placed in a bin assigned to it (on top of items previously placed in that bin) as soon as it arrives. This placement cannot be changed later. In addition, the following LIB[3] constraint should be obeyed for any used bin:

$$[i \text{ is below } j \text{ in a used bin}] \implies [a_i \geq a_j]. \tag{16}$$

[3]LIB stands for longest item at the bottom.

A feasible solution is one where the sum of the item sizes in each used bin is at most equal to the bin size. The available bin sizes consist of a finite set $\mathcal{B} = \{s_j : 1 \leq j \leq K\}$, $s_j < s_{j+1}$, $1 \leq j \leq K-1$. The bin sizes are normalised, that is, s_K (the largest bin size) is equal to one. The smallest bin size s_1 is greater than zero. The goal is to find a feasible solution that minimises the sum of the size of used bins.

The online condition essentially reduces to the following *Online Constraint*: In a used bin, if item i is below item j, then i should have arrived prior to j in the input list L, that is,

$$[i \text{ is below } j \text{ in a used bin}] \implies [i < j]. \tag{17}$$

In a uniform-sized bin packing problem, $s_1 = s_K = 1$, that is, all bins are of unit size.

The approximation complexity of a problem P is usually measured by the *approximation ratio* that a heuristic H for P can guarantee, over all instances of P. The approximation ratio $R_H(I)$ obtained by H for a given instance I of P is given by

$$R_H(I) = \frac{\text{value obtained by } H \text{ on } I}{\text{value of an optimal solution for } I} \tag{18}$$

3.1 Bin Packing is Polynomially Bound

We now show that the variable size bin packing problem, online or offline, with or without the LIB constraint, is polynomially bound.

Optimal solution opt(I) for an instance I. An upper bound on the sum of the sizes of bins used, occurs when each item is placed in its own bin. For an instance I (a list L with n items, and a collection of item sizes),

$$opt(I) \leq \sum_{i=1}^{n} \lceil a_i \rceil_{\mathcal{B}}. \tag{19}$$

Note that $\lceil a_i \rceil_{\mathcal{B}}$ is a certain bin size[4] in $\mathcal{B}$. Suppose we call this bin size as θ_i to denote that this bin holds item i, and hence $\theta_i \in \mathcal{B}$. Since $\theta_i \leq 1$, the representation of θ_i in a Turing machine (TM) in binary form will require $\lceil 1 + \log(1/\theta_i) \rceil$ bits. The reason is, for a number $x \in (0, 1]$, the lower the number, the more bits are required to represent it — it needs $\log(1/x)$ bits, plus an additional bit to denote that $x \leq 1$ (otherwise the $\log(1/x)$ bits will be interpreted as representing x^{-1}).

Thus for the n items in total, the number of bits required to represent this in binary form in a Turing machine will be

$$\sum_{i=1}^{n} \left\lceil 1 + \log \frac{1}{\theta_i} \right\rceil = n + \sum_{i=1}^{n} \left\lceil \log \frac{1}{\theta_i} \right\rceil. \tag{20}$$

Size of an instance I. To represent an instance of variable-size bin packing, we need to represent the n item sizes and K bin sizes. Note that all these entities are at most equal to one. As per the argument above, each item size a_i needs $\lceil 1 + \log(1/a_i) \rceil$ bits to be represented

[4] $\lceil a_i \rceil_{\mathcal{B}}$ is the bin size in $\mathcal{B}$ that is closest to a_i and at least as much as a_i. For example, if $a_i = 0.7$ and $\mathcal{B}$ = {0.2, 0.4, 0.6, 0.8, 1.0}, then $\lceil a_i \rceil_{\mathcal{B}} = 0.8$.

in binary, and each bin size $s_j \in \mathcal{B}$ needs $\lceil 1 + \log(1/s_j) \rceil$ bits. Thus the number of bits needed to represent an instance of our problem will be

$$\sum_{i=1}^{n} \left\lceil 1 + \log \frac{1}{a_i} \right\rceil + \sum_{j=1}^{K} \left\lceil 1 + \log \frac{1}{s_j} \right\rceil = n + K + \sum_{i=1}^{n} \left\lceil \log \frac{1}{a_i} \right\rceil + \sum_{j=1}^{K} \left\lceil \log \frac{1}{s_j} \right\rceil . \tag{21}$$

Now compare (20) and (21). For each $i \in [1, n]$, $0 < a_i \leq \theta_i$. Thus

$$\left\lceil \log \frac{1}{\theta_i} \right\rceil \leq \left\lceil \log \frac{1}{a_i} \right\rceil \implies \sum_{i=1}^{n} \left\lceil \log \frac{1}{\theta_i} \right\rceil \leq \sum_{i=1}^{n} \left\lceil \log \frac{1}{a_i} \right\rceil \tag{22}$$

From this, it follows that in binary form, the optimal solution value is less than the size of the instance — and hence

Lemma 6. *The optimal solution value is less than a polynomial in the size of the instance I. Or,*

$$opt(I) \leq |I| \leq p(|I|) \tag{23}$$

for the variable size bin packing problem, online or offline, with or without the LIB constraint.

Hence the proof of the polynomial bound.

3.2 Previous Work

Optimisation problems can be classified in three different ways: syntactic classes, computational classes, and approximation classes.

Problems in a certain *syntactic class* share a common property of logical expressibility — for example, the ability to express the first order part in PNF as a Σ_2 formula.

Problems in a certain *approximation class* share a property in terms of how well solutions can be approximated by polynomial-time heuristics — for example, the APX class of optimisation problems where it is possible to obtain a solution whose value is guaranteed to be within a constant factor of an optimal solution value (using polynomial time heuristics).

Problems in a certain *computation class* share a property in terms of how fast an exact solution can be obtained, or how fast a solution with a certain approximation guarantee can be obtained — for example, the class of optimisation problems that have polynomial time algorithms which return an optimal solution in linear time.

Problems in a certain syntactic class $\mathcal{S}$ and a certain computational class $\mathcal{C}$ can overlap. In certain cases, the overlap is exact, as in the case of the computational class **NP** which is precisely the syntactic class of problems expressible in second-order existential logic, also known as ESO [Fag74].

Similarly, the syntactic class MAXSNP (maximisation problems that can be defined by formulae with a quantifier-free first order part) contains exactly those problems that can be approximated to within a constant factor of the optimal value using polynomial time heuristics [KMSV98].

Next, we consider how to express optimisation problems syntactically. Consider these examples of polynomially bound problems:

(1) Minimising the number of edges (SHORTEST PATH problem with unit weight edges): Each edge in the shortest path is a 2-tuple. Thus we minimise the number of 2-tuples that obey a certain condition (that is, being on a path from origin to destination).

(2) Maximising the number of vertices that form a complete subgraph (MAX CLIQUE): Each vertex in a solution (the clique) is a unary tuple (single vertex). Thus we maximise the number of unary tuples that form a clique.

(3) Maximising the number of satisfiable clauses (MAX3SAT): each clause that is true is a 3-tuple. We maximise the number of 3-tuples that obey the condition that the 3-tuple be a disjunction of literals, at least one of which is true.

Motivated by such observations as above, Papadimitriou and Yannakakis [PY91] proposed the class MAXNP of maximisation problems whose optimal values can be defined as

$$opt_Q(\mathbf{A}) = \max_{\mathbf{S}} |\{\mathbf{w} : (\mathbf{A}, \mathbf{S}) \models \phi\}| = \max_{\mathbf{S}} |\{\mathbf{w} : (\mathbf{A}, \mathbf{S}) \models \exists \mathbf{y}\ \psi(\mathbf{w}, \mathbf{y}, \mathbf{S})\}|. \tag{24}$$

where $\mathbf{y} = (y_1, y_2, \cdots, y_q)$, $\mathbf{w} = (w_1, w_2, \cdots, w_k)$, ψ is a quantifier-free formula, and $\mathbf{S}$ is defined in Definition 4. They showed that every problem that can be characterised in this manner is in APX (defined earlier in this subsection).

On the other hand, there are also negative results about certain syntactic classes. For the class MAXSNP (which differs from MAXNP in that the first order part is free of all quantifiers), Arora and Safra [ALM+98] showed that no (maximisation) problem in this class has a polynomial time approximation scheme unless P=NP.

For our purposes, we will use a 1995 result from Kolaitis and Thakur [KT95]. A few definitions first.

Definition 7. *[KT95] For each $n \geq 1$, let* MINFΠ_n *(F means feasible) be the class of minimisation problems Q, whose optimal values on finite structures $\boldsymbol{A}$ over a vocabulary σ are defined as*

$$\begin{aligned} opt_Q(\mathbf{A}) &= \min_{\mathbf{S}} |\{\mathbf{w} : (\mathbf{A}, \mathbf{S}) \models \phi(\mathbf{w}, \mathbf{S})\}|, \ \textit{if } \exists \mathbf{S} \ni (\mathbf{w}, \mathbf{A}, \mathbf{S}) \models \phi(\mathbf{w}, \mathbf{S}) \\ &= |A|^m, \ \textit{otherwise}, \end{aligned} \tag{25}$$

where $\phi()$ is a Π_n first order sentence, m is the arity of $\mathbf{w}$, and A is the universe of structure $\mathbf{A}$. A sentence is a formula that has no free variables.

The MINFΣ_n *class of problems is defined similarly, except that in this case, $\phi()$ is a Σ_n first order sentence.*

MINF$^+\Pi_n$ *(*MINF$^+\Sigma_n$*) is similar to the* MINFΠ_n *(*MINFΣ_n*) class except that each occurence of each of the predicates $S_i \in \mathbf{S}$ $(1 \leq i \leq p)$ in $\phi(\mathbf{w}, \mathbf{S})$ is positive*[5].

MINF$^+\Pi_n(k)$ *is similar to* MINF$^+\Pi_n$*, except that the second-order predicate (see Def. 4) whose cardinality we wish to minimise appears at most k times in each clause of ϕ.*

[5]Example: In the expression $S_1(x_1, x_2) \vee \neg S_2(x_3, x_4, x_5)$, the predicate S_1 appears positively and S_2 occurs negatively.

3.3 Syntactic Expression for Uniform Size Bin Packing

In this subsection, we provide an expression for the optimal solution value for the online USBP (uniform size bin packing) problem with LIB constraints, using second order Logic — and thus show that this problem is in the $\text{MINF}^{+}\Pi_2$ class. All bins are of unit size.

3.3.1 Variables and Universes

There are two distinct types of entities: index numbers (for items and bins), and sizes (for items). It is convenient to use *Two Sorted Logic*, implying there will be two universes in our structure $\mathbf{A}$ representing an instance of bin packing — one for the index numbers of items and bins (U_N), and one for item sizes (U_I).

Assume that n bins, numbered 1 to n, are available. The objective is to choose the least number of bins from the set $\{i \mid 1 \leq i \leq n\}$. (The maximum number of bins we will ever need is n.)

U_N could simply be a subset of the Natural numbers $\mathbf{N}$. Thus

$U_N = \{0, 1, 2, 3, \cdots n\}$, where n is the number of items in the input list L.

$U_I = \{a_1, a_2, \cdots, a_n\}$ (the given item sizes).

From a (sufficiently large) variable list V, we need (first order) predicate symbols to decide which universe a variable x belongs to. We shall describe these and other predicates next.

3.3.2 First Order Predicates and Functions

First order (FO) predicates describe properties of an instance (which is represented by a structure $\mathbf{A}$) of an optimisation problem. FO predicate symbols are part of the FO vocabulary σ. We will use the following FO predicates:

$C(x)$: True iff variable x assumes a value from U_N.

$PLUS(x_1, x_2, x_3)$: As defined in Section 2.4.

In addition, we need a unary function *size* to map item numbers to their sizes.

3.3.3 Second Order Predicates

We will use the following second-order (SO) predicates — these are not part of the vocabulary σ. From an optimisation viewpoint, SO predicates are necessary to describe *solutions* to a *problem instance*, not instances themselves.

$BELOW(x, y)$: True iff item x is below item y in a bin.

$IN(x, y)$: True iff item x is placed in bin y.

$S(w)$: True iff bin w ($1 \leq w \leq n$) is *used*.

We want to minimise the number of bins used. Each bin can be represented by a unary tuple w, hence we will minimise the number of such tuples.

3.3.4 Constraints

We are now ready to describe expressions for the problem constraints. All expressions below are *sentences*, that is, expressions with no free variables.

(a) If two items x and y are placed such that x is below y, then they should be in the same bin (and that bin should be used), x should have arrived before y, and x should be at least as long as y:

$$\begin{aligned}\phi_1' \equiv \forall x \forall y\ BELOW(x,y) \quad \longrightarrow \quad & \exists w\ S(w) \wedge IN(x,w) \wedge IN(y,w) \wedge (x < y) \\ & \wedge\ size(x) \geq size(y) \wedge C(x) \wedge C(y) \wedge C(w).\end{aligned}$$

The above can be rewritten without the implication[6] as

$$\begin{aligned}\phi_1 \equiv\ & \forall x \forall y\ \exists w\ \tau_1, \text{ where,} \\ \tau_1 \equiv\ & ABOVE(x,y) \vee [C(x) \wedge C(y) \wedge C(w) \wedge \\ & S(w) \wedge IN(x,w) \wedge IN(y,w) \wedge (x < y) \wedge\ size(x) \geq size(y)], \\ ABOVE(x,y) \equiv\ & \neg BELOW(x,y).\end{aligned} \tag{26}$$

Note that $ABOVE(x,y)$ includes the possibility that x and y may be in different bins.

(b) Every item x should be placed in some bin w, and that bin should be *used*:

$$\phi_2' \equiv \forall x\ \exists w\ C(x) \ \longrightarrow\ C(w) \wedge IN(x,w) \wedge S(w).$$

Rewrite the above without the implication as

$$\phi_2 \equiv \forall x\ \exists w\ \tau_2, \text{ where } \tau_2 \equiv \neg C(x)\ \vee\ [C(w) \wedge IN(x,w) \wedge S(w)]. \tag{27}$$

Henceforth, we shall omit the $C(x)$ terms from expressions — it should be clear from the context the variables for which this holds true. Since $C(x)$ is a first-order predicate, it will not affect the approximability result (which is based on the signs of second-order predicates only).

3.3.5 Bin Capacity Constraints

The constraints in this subsection enforce the condition that the sum of the sizes of items in each of the n bins is at most one. Introduce a new set of variables $z_{x,u}$ of type ITEM SIZE and assign values as below. For every item x and every bin u, $z_{x,u}$ takes the value of either zero or $size(x)$, the size of item x:

$$\phi_3 \equiv \forall x\ \forall u\ \exists z_{x,u}\ (z_{x,u} = size(x)) \vee (z_{x,u} = 0). \tag{28}$$

If item x is placed in bin u, then assign the size of x to $z_{x,u}$, and zero otherwise. The following constraint, which expands (28), makes an assignment of $size(x)$ to $z_{x,u}$ when x is placed in u, or an assignment of zero in general:

$$\phi_3 \equiv \forall x\ \forall u\ \ \exists z_{x,u}\ [IN(x,u) \wedge z_{x,u} = size(x)] \vee [z_{x,u} = 0]. \tag{29}$$

[6]Recall that $A \longrightarrow B$ is the same as $\neg A \vee B$. Furthermore, $\neg(A \wedge B)$ is the same as $(\neg A) \vee (\neg B)$.

However, (29) is clearly insufficient. It could assign a zero value to $z_{x,u}$ even when x is in u. When $IN(x,u)$ is true, only $size(x)$ can be assigned to $z_{x,u}$. This is enforced by constraint (30), which says that the sum of the sizes of all items in all bins is equal to a certain constant T, which is derived from the input instance $\mathbf{A}$. Let

$$\psi_1 \equiv PLUS(z_{1,1}, z_{1,2}, \alpha_{1,1}) \wedge PLUS(\alpha_{1,1}, z_{1,3}, \alpha_{1,2}) \wedge \cdots \wedge PLUS(\alpha_{1,n-2}, z_{1,n}, \beta_1),$$

$$\psi_2 \equiv PLUS(z_{2,1}, z_{2,2}, \alpha_{2,1}) \wedge PLUS(\alpha_{2,1}, z_{2,3}, \alpha_{2,2}) \wedge \cdots \wedge PLUS(\alpha_{2,n-2}, z_{2,n}, \beta_2),$$

$$\vdots$$

$$\psi_n \equiv PLUS(z_{n,1}, z_{n,2}, \alpha_{n,1}) \wedge PLUS(\alpha_{n,1}, z_{n,3}, \alpha_{n,2}) \wedge \cdots \wedge PLUS(\alpha_{n,n-2}, z_{n,n}, \beta_n),$$

$$\tau_3 \equiv \psi_1 \wedge \cdots \wedge \psi_n \wedge (\beta_1 \leq 1) \wedge \cdots \wedge (\beta_n \leq 1),$$

$$\tau_4 \equiv PLUS(\beta_1, \beta_2, \gamma_1) \wedge PLUS(\gamma_1, \beta_3, \gamma_2) \wedge \cdots \wedge PLUS(\gamma_{n-2}, \beta_n, T),$$

where $T = \sum_{i=1}^{n} size(i)$ = sum of the sizes of all items in the input list. Then

$$\tau_5 \equiv \exists\alpha_{1,1} \cdots \exists\alpha_{n,n-2}\ \exists\beta_1 \cdots \exists\beta_n\ \exists\gamma_1 \cdots \exists\gamma_{n-2}\ \tau_3 \wedge \tau_4. \tag{30}$$

Observe that τ_3 in constraint (30) enforces the bin capacity of one unit for each bin — the variable β_i is the sum of the sizes of items in bin i. The expression for τ_4 asserts that the sum of the sizes of all items in all bins should equal to T. Each ψ_i adds the item sizes in bin i.

Since τ_3, τ_4 and τ_5 make use of the z's defined in ϕ_3, we can then expand ϕ_3 to include τ_5:

$$\begin{aligned} \phi_3 &\equiv \forall x\ \forall u\ \ \exists z_{x,u}\ \{[IN(x,u) \wedge z_{x,u} = size(x)] \vee [z_{x,u} = 0]\} \wedge \tau_5 \\ &\equiv \forall x\ \forall u\ \exists z_{x,u}\ \exists\alpha_{1,1} \cdots \exists\alpha_{n,n-2}\ \exists\beta_1 \cdots \exists\beta_n\ \exists\gamma_1 \cdots \exists\gamma_{n-2} \\ &\quad [IN(x,u) \wedge z_{x,u} = size(x) \wedge \tau_3 \wedge \tau_4] \vee [z_{x,u} = 0 \wedge \tau_3 \wedge \tau_4]. \end{aligned} \tag{31}$$

In other words,

$$\begin{aligned} \phi_3 &\equiv \forall x\ \forall u\ \exists z_{x,u}\ \exists\alpha_{1,1} \cdots \exists\alpha_{n,n-2}\ \exists\beta_1 \cdots \exists\beta_n\ \exists\gamma_1 \cdots \exists\gamma_{n-2}\ \tau_6 \vee \tau_7, \text{ where} \\ \tau_6 &\equiv [IN(x,u) \wedge z_{x,u} = size(x) \wedge \tau_3 \wedge \tau_4] \text{ and} \\ \tau_7 &\equiv [z_{x,u} = 0 \wedge \tau_3 \wedge \tau_4]. \end{aligned} \tag{32}$$

Since all constraints should be obeyed for a feasible solution, we have

$$\begin{aligned} \phi &\equiv \phi_1 \wedge \phi_2 \wedge \phi_3 \equiv \\ &\forall x \forall y \forall u\ \exists w \exists z_{x,u} \exists\alpha_{1,1} \cdots \exists\alpha_{n,n-2}\ \exists\beta_1 \cdots \exists\beta_n\ \exists\gamma_1 \cdots \exists\gamma_{n-2}\ \tau_1 \wedge \tau_2 \wedge (\tau_6 \vee \tau_7). \end{aligned} \tag{33}$$

Note that the expression ϕ is in Π_2 form. It is a sentence since all variables are bound.

3.3.6 Optimal Solution Value and DNF

The optimal solution value to an instance $\mathbf{A}$ of uniform size bin packing (USBP) is given by

$$opt_{USBP}(\mathbf{A}) = \min_{\mathbf{S}} |\{w : (\mathbf{A}, \mathbf{S}) \models \phi\}| = \min_{\mathbf{S}} |\{w : (\mathbf{A}, \mathbf{S}) \models \phi(w, \mathbf{S})\}|, \tag{34}$$

where w is the operand of S occuring in (26) and (27). The sequence $\mathbf{S}$ consists of the three second-order predicates (S, IN, $ABOVE$).

The task at hand now is to put expressions ϕ_1 through ϕ_3 together in DNF (disjunctive normal form), in place of the CNF (conjunctive normal form) above. For example, the CNF expression

$$(a_1 \vee a_2) \wedge (b_1 \vee b_2) \wedge (c_1 \vee c_2) \tag{35}$$

can be written in DNF[7] as

$$(a_1 \wedge b_1 \wedge c_1) \vee (a_1 \wedge b_1 \wedge c_2) \vee (a_1 \wedge b_2 \wedge c_1) \vee (a_1 \wedge b_2 \wedge c_2) \vee \cdots \vee (a_2 \wedge b_2 \wedge c_2). \tag{36}$$

Given the example in (35-36), it is not difficult to convert ϕ (in CNF) to its DNF ϕ'. However, the crucial point to note is that, during such a conversion, second-order predicates do not change signs. In particular, the predicate S, whose cardinality we wish to optimise, remains positive. All three second-order (SO) predicates appear positively in ϕ and ϕ'.

The only SO predicate that could possibly appear more than once in a clause of ϕ' is $S(w)$ — however, it appears positively in the clause, as $S(w) \wedge S(w)$, when we seek to "integrate" τ_1 and τ_2 — note that $S(w)$ appears in (26) and (27) only. Naturally, $S(w) \wedge S(w)$ can be reduced to $S(w)$, and hence we can say that this predicate appears only once in each clause of ϕ'.

Thus, this problem falls in the MINF$^+\Pi_n(1)$ class defined in Definition 7. We now refer to a result from [KT95] on problems belonging to this class:

Theorem 7. *For an instance $\boldsymbol{A}$ to a problem Q in the class* MINF$^+\Pi_2(k)$*, an approximate solution within $c[opt_Q(\mathbf{A})]^k \log|A|$ can be guaranteed, where $|A|$ is the size of the universe of $\boldsymbol{A}$, and c is a constant. In particular, problems in* MINF$^+\Pi_2(1)$ *are $\log|A|$-approximable.*

Hence we can conclude that

Theorem 8. *The uniform size bin packing problem with LIB constraints is approximable within a $\Theta(\log n)$ factor of the optimal solution value, where n is the number of items in the input list.*

4 Future Research

Most of this paper studies only maximisation problems — research should be carried out for minimisation problems as well, and the connection between the two (for example, using the duality concept in mathematical programming). Complete problems should be discovered for the respective subclasses.

Since the decision version of the weighted MAXFLOW problem (where arc capacity can be any non-negative integer) is complete for the class $\mathbf{P}$ [GHR95, Imm99], the optimisation version of weighted MAXFLOW is likely to be a complete problem for $\mathbf{P'}$ — this is yet to be proven.

[7] If the CNF expression consists of n clauses and the clauses contain l_1, l_2, $\cdots$, l_n number of literals respectively, then the number of clauses in DNF is equal to $\Pi_{i=1}^{n} l_i$.

Acknowledgments

We benefited from discussions with the theoretical computer science group at the University of Leicester (UK), as well as with J. Radhakrishnan and A. Panconesi at TIFR (Mumbai). The proof in Section 2.1 is due to Radhakrishnan [Rad05]. A preliminary version of the paper was presented at the annual meeting of the Australian Mathematical Society at Perth (Western Australia) in September 2005.

References

[ALM+98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof Verification and Intractability of Approximation Problems. *Journal of the ACM*, 45(3):501–555, 1998.

[E. 91] E. Grädel. The expressive power of second order Horn logic. In *STACS 1991: Proceedings of the 8th annual symposium on Theoretical aspects of computer science — Lecture Notes in Computer Science 280*, pages 466–477. Springer-Verlag, 1991.

[Fag74] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R. Karp, editor, *Complexity of Computations*, pages 43–73. SIAM-AMS Proceedings (no.7), 1974.

[GHR95] R. Greenlaw, H. James Hoover, and W.L. Ruzzo. *Limits to Parallel Computation: P-Completeness Theory.* Oxford University Press, 1995.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman (New York), 1979.

[Hå97] J. Håstad. Some Optimal Inapproximability Results. In *ACM-STOC 1997: Proceedings of the 29th ACM Symposium on the Theory of Computing*, pages 1–10, 1997.

[Imm99] Neil Immerman. *Descriptive Complexity.* Springer-Verlag, 1999.

[KKM94] R. Kohli, R. Krishnamurti, and P. Mirchandani. The Minimum Satisfiability Problem. *SIAM Journal of Discrete Mathematics*, 7:275–283, 1994.

[KMSV98] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On syntactic versus computational views of approximability. *SIAM Journal of Computing*, 28(1):164–191, 1998.

[KT94] P.G. Kolaitis and M.N. Thakur. Logical Definability of NP-Optimisation Problems. *Information and Computation*, 115(2):321–353, December 1994.

[KT95] P.G. Kolaitis and M.N. Thakur. Approximation Properties of NP-Minimisation Problems. *Journal of Computer and System Sciences*, 50:391–411, 1995.

[Man] P. Manyem. Syntactic Characterisations of Polynomial-Time Optimisation Classes. Submitted to a journal. Available at: `http://uob-community.ballarat.edu.au/~pmanyem/syntax-one.pdf`.

[PR93] A. Panconesi and D. Ranjan. Quantifiers and approximation. *Theoretical Computer Science*, 107:145–163, 1993.

[PY91] C.H. Papadimitriou and M. Yannakakis. Optimization, Approximation, and Complexity Classes. *Journal of Computer and System Sciences*, 43(3):425–440, December 1991.

[Rad05] J. Radhakrishnan. Personal communication, December 2005.

[Zim98] M. Zimand. Weighted NP-Optimisation Problems: Logical Definability and Approximation Properties. *SIAM Journal of Computing*, 28(1):36–56, 1998.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/