Electronic Colloquium on Computational Complexity, Report No. 163 (2024)

On defining $\mathcal{PPT}$-search problems and $\mathcal{PPT}$-optimization problems

Roei Tell *

October 24, 2024

Abstract

This note revisits the study of search problems that are solvable in probabilistic polynomial time. Previously, Goldreich (2011) introduced a class called "$\mathcal{BPP}$-search", and studied search-to-decision reductions for problems in this class.

In Goldreich's original formulation, the definition of what counts as "successfully solving" a $\mathcal{BPP}$-search problem is implicit, and this opens the door to multiple possible interpretations. We suggest two approaches:

1. We propose **our preferred definition** for the notion of solving $\mathcal{BPP}$-search problems. The resulting class of search problems is useful, since it captures natural problems of interest, and since it has valuable technical properties. However, the a-priori rationale for studying the class is not fully satisfying.
2. We propose an **alternative formulation** of the class of search problems solvable in ppt, which is focused on optimization (i.e., finding a solution of best quality). The resulting class has a clearer meaning, and it turns out to be **computationally equivalent** to the class resulting from our aforementioned definition of $\mathcal{BPP}$-search (under deterministic polynomial-time reductions).

The optimization-based definition seems cleaner and more appealing. However, since the two classes above are computationally equivalent, it can be useful to use whichever definition is technically more convenient in the relevant context.

On the companion paper. This paper presents my perspective on a joint project conducted together with Oded Goldreich, whereas Oded's perspective is presented in the companion paper [Gol24]. The technical contents of both papers has significant overlap, and both papers propose the same definition that is referred to in the current text as $\mathcal{PPT}$-optimization (i.e., Definition 5).

*The University of Toronto, Department of Computer Science. Email: roei@cs.toronto.edu

ISSN 1433-8092

Contents

1 Introduction

This note revisits the study of the class of search problems that are solvable in probabilistic polynomial time, which was introduced by Goldreich [Gol11] as "$\mathcal{BPP}$-search". As usual in complexity theory, search problems are arguably more natural than decision problems, and our focus on the latter is typically justified by a search-to-decision reduction. Accordingly, a technical result in [Gol11] studied reductions of $\mathcal{BPP}$-search problems to $pr\mathcal{BPP}$ (i.e., to decision promise-problems).

The current text is motivated by the observation that the original definition and result did not fully resolve the issue. Specifically, the notion of **successfully solving a $\mathcal{BPP}$-search problem** is addressed there in an implicit way, and this opens the door to multiple interpretations. Unfortunately, none of the definitions for this notion that we are aware of seems fully satisfactory: Some definitions yield a class that does not support a search-to-decision reduction, or that contains infeasible search problems, whereas other definitions have conceptual shortcomings.

The contribution of this text is to propose two approaches for handling this challenge. First, we **suggest our preferred definition** for the notion of successfully solving $\mathcal{BPP}$-search problems. In Section 2 we present this suggestion, and demonstrate that the definition yields a useful class of search problems. In particular, we show that the resulting class captures natural problems of interest, and has several technical properties we would like such a class to have (e.g., a search-to-decision reduction). We also point out the conceptual shortcomings of this interpretation.

Secondly, we suggest an **alternative formulation** for the class of search problems solvable in ppt, via a definition that is focused on optimization (i.e., on finding a solution of best quality). This alternative formulation has a clearer meaning, and the resulting class of problems is computationally equivalent to the class given by our first approach (i.e., solving any problem from one class reduces in deterministic polynomial time to solving a problem from the other class). In Section 3 we present this alternative formulation, the equivalence result, and some technical properties of the class.

Lastly, in Appendix A we discuss other possible formulations, and explain some of the issues leading us to view these formulations less favorably.

Notation. Throughout the text, we will follow some of the notation and terminology introduced in the companion paper by Goldreich [Gol24]. In particular, we will refer to $\mathcal{BPP}$-search problems as "$\mathcal{PPT}$-search" problems, and we will use similar notations for the new optimization-based formulation that is presented in Section 3.

A quick comparison with the companion paper. The second formulation that we suggest is identical to the main suggestion in Goldreich's companion paper [Gol24], and both texts suggest this formulation as an appealing alternative to the original definition from [Gol11]. However, the first formulation that we suggest is not presented as an appealing alternative in the companion paper, due to conceptual reasons that are explained in [Gol24, Section 2.3]. Also, while the technical results underlying the computational equivalence between the two formulations appear in [Gol24], they are not interpreted there as yielding an equivalence.

2 The definition from [Gol11] and the challenges it presents

Recall that a standard approach for specifying a search problem is to use a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$, where a pair (x,y) represent an instance x to the problem and a candidate solution y for the instance x. The fact that $(x,y) \in R$ means that y is a valid solution for x, and the search task is "given x, output y such that $(x,y) \in R$, if it exists". For any instance x, we define the set of solutions $R(x) = \{y : (x,y) \in R\}$.

2.1 Modeling search problems as promise problems

Motivated by the generic appeal of promise problems, and by the desire to present a more general definition (that would capture more efficiently solvable search problems), Goldreich [Gol11] suggested considering **promise problems** in this context. A first and more standard step is to consider a promise on the *instances* to the problem, whose meaning is as usual: Some instances are irrelevant and will never be encountered (or can be efficiently recognized and discarded). The less standard step was to also consider a **promise on candidate solutions** to the problem.

Specifically, in this formulation, a search problem is specified by two sets of instance-solution pairs $\Pi = (\Pi^Y, \Pi^N)$ where $\Pi^Y, \Pi^N \subseteq \{0,1\}^* \times \{0,1\}^*$ and $\Pi^Y \cap \Pi^N = \emptyset$. Then, any fixed instance x yields a tripartition of the potential solutions y: There are solutions $y \in \Pi^Y(x)$,[1] which we think of as good and valid solutions; and solutions $y \in \Pi^N(x)$, which we think of as bad and invalid solutions; and solutions y "in limbo" such that $y \notin \Pi^Y(x) \cup \Pi^N(x)$, whose meaning is less clear.

Thus, whenever we specify a search problem as a promise problem (i.e., regardless of whether or not the problem lies in some complexity class), we encounter two related questions: What is the meaning of the tripartition, and in particular what is the meaning of the solutions in limbo; and **what do we define as solving the problem**, and in particular whether limbo solutions count as valid ones in this context.

Two notions of "solving". For a search problem $\Pi = (\Pi^Y, \Pi^N)$, we say that a search algorithm strongly solves Π if, when given x such that $\Pi^Y(x) \neq \emptyset$ it outputs $y \in \Pi^Y(x)$; and we say that a search algorithm weakly solves Π if, when given x such that $\Pi^Y(x) \neq \emptyset$ it outputs y satisfying the weaker requirement $y \notin \Pi^N(x)$.[2]

2.2 The original definition, and what needs to be explicit in it

Intuitively, we wish to focus on problems wherein solutions can be both efficiently found and efficiently recognized, where "efficiently" in this context means probabilistic polynomial time. (Problems in which solutions cannot be efficiently recognized

[1] Recall that for $\Pi = (\Pi^Y, \Pi^N)$ we define $\Pi^Y(x) = \{y : (x,y) \in \Pi^Y\}$ and $\Pi^N(x) = \{y : (x,y) \in \Pi^N\}$.

[2] Indeed, these definitions pose no requirements of the search algorithm (whether it is weak or strong) when it is given x such that $\Pi^Y(x) = \emptyset$. If the decision problem of Π can be solved efficiently (i.e., an efficient algorithm accepts Π^Y and rejects Π^N), then we may assume wlog that any algorithm that weakly solves Π always outputs either a valid solution (i.e., $y \notin \Pi^N(x)$) or $\perp$.

may still be interesting, but defining corresponding classes requires more care; see discussion of "the meaningfulness principle" in [Gol24, Section 3.3].)

Combining the two requirements, we obtain the original definition from [Gol11, Definition 3.2], to which we will refer as "$\mathcal{PPT}$-search" (following [Gol24]):

Definition 1 ($\mathcal{PPT}$-search, originally called "$\mathcal{BPP}$-search")**.** *We say that* $\Pi = (\Pi^{\mathsf{Y}}, \Pi^{\mathsf{N}})$ *is in* $\mathcal{PPT}$*-search if the following two conditions hold:*

1. *There is a ppt algorithm* V *that solves the decision problem of* Π *(i.e., accepts pairs in* Π^{Y} *and rejects pairs in* Π^{N}*).*
2. *There is a ppt algorithm* F *that, when given* x *such that* $\Pi^{\mathsf{Y}}(x) \neq \emptyset$*, satisfies* $\Pr_r[F(x,r) \in \Pi^{\mathsf{Y}}(x)] \geq 2/3$.

Note that the algorithm in Item (2) of Definition 1 strongly solves the problem; that is, for a problem to be in $\mathcal{PPT}$-search, we require the existence of an algorithm strongly solving the problem. However, this definition does not explicitly specify what is the meaning of the tripartition and of limbo solutions, and in particular does not specify **what do we define as solving** the search problem (i.e., weakly solving or strongly solving). This is important, since a main question we are interested in is "what is the complexity of solving $\mathcal{PPT}$-search problems" (and specifically, whether all $\mathcal{PPT}$-search problems are solvable in deterministic polynomial time).

The requirement in Item (2) suggests that solving a $\mathcal{PPT}$-search problem may be defined as strongly solving. This is a natural interpretation of the intention of the original paper, and it is the one intended by Goldreich (as indicated in personal communication). However, the search-to-decision in [Gol11, Theorem 3.5] only yields an algorithm that weakly solves the problem,[3] which suggests another interpretation: namely, that we may also settle for weakly solving the problem.

2.3 Our suggested definition: Solving means weakly solving

One may try to provide explicit definitions now, although neither option seems ideal to us. One option is to think of limbo solutions as inadmissible, and only allow search algorithms that strongly solve the problem. However, in this case it is not clear why we allow the verification algorithm V in Item (1) of Definition 1 to accept inadmissible limbo solutions. In fact, since V may accept limbo solutions, some search problems in the resulting class cannot be solved deterministically in *any* running time (i.e., this does not align with our intention of formulating a class wherein solutions can be verified efficiently, and hence found by brute-force). See Appendix A.1 for details.

Our suggested definition is to think of limbo solutions as adequate, although not as good as solutions y such that $(x,y) \in \Pi^{\mathsf{Y}}$, and define that **solving the problem means weakly solving it**. Note that the requirement in Item (2) is still intact: That is,

[3] The resulting algorithm is not presented in [Gol11] as weakly solving the problem, but rather as strongly solving a related problem. Nevertheless, by inspecting the definition of the related problem, what we obtain is indeed an algorithm that weakly solves the original problem.

from this view, solving a problem means finding y of quality that is adequate or better, but a problem lies within the complexity class defined in Definition 1 only if there is a ppt algorithm meeting the stricter requirement of finding y of good quality.

The reason for suggesting this definition is that the resulting complexity class is useful: It captures important search problems that we are interested in, and it has valuable technical properties that we would like a complexity class to have.

Technical results. To demonstrate the foregoing assertion, let us first present some natural (and very general) search problems that are in $\mathcal{PPT}$-search. We stress that the following problems belong to this class regardless of how we define solving (i.e., weakly solving or strongly solving). However, if we define solving as weakly solving, then $pr\mathcal{BPP} = pr\mathcal{P}$ implies that these problems can be solved in deterministic polynomial time (in fact, this is true for all $\mathcal{PPT}$-search problems – see Theorem 3).

Proposition 2 (interesting problems in $\mathcal{PPT}$-search; informal, see Proposition 12)**.** *The following problems all belong to the class outlined in Definition 1:*

1. **(Estimating the acceptance probability of a circuit.)** *Given a circuit $C\colon \{0,1\}^n \to \{0,1\}$ and an accuracy value 1^t, estimate $\Pr_r[C(r) = 1]$, where estimations with accuracy $1/t$ or better are good and estimations with accuracy $2/t$ or worse are bad.*

2. **(Finding a fooling set for a circuit.)** *Given a circuit $C\colon \{0,1\}^n \to \{0,1\}$ and an accuracy value 1^t, output a set $S \subseteq \{0,1\}^n$ such that $\Pr_{s\in S}[C(s) = 1]$ approximates $\Pr_{r\in\{0,1\}^n}[C(r) = 1]$. Sets S with approximation of $1/t$ or better are good, sets S with approximation of $2/t$ or worse are bad.*

3. **(Polynomial identity testing.)** *Given an n-variate arithmetic circuit C over a finite field $\mathbb{F}$, output $x \in \mathbb{K}^n$ such that $C(x) \neq 0$, where $\mathbb{K} \supseteq \mathbb{F}$ may be an extension of $\mathbb{F}$.*

For all search problems above, if $pr\mathcal{BPP} = pr\mathcal{P}$ then there is a deterministic polynomial-time algorithm that (weakly) solves the problem.

The proof of Proposition 2 appears in Section B. Next, we state some technical properties of $\mathcal{PPT}$-search, when solving is defined as weakly solving. Importantly, this class indeed supports a search-to-decision reduction (i.e., all problems in the class are reducible to $pr\mathcal{BPP}$). In addition, the class has a natural (total) complete problem, supports error-reduction for the search algorithm, and has a time hierarchy result.[4]

Theorem 3 (technical properties of our interpretation of $\mathcal{PPT}$-search)**.** *Consider the class of $\mathcal{PPT}$-search problems wherein solving the problem means* **weakly solving***. Then, the following properties hold:*

1. **(Complete problem.)** *There is a total $\mathcal{PPT}$-search problem denoted* APEP *such that solving any $\mathcal{PPT}$-search problem reduces in deterministic polynomial time to solving*

[4]These properties are not trivial for randomized classes. For context, recall that in the study of decision problems, a complete problem and a time hierarchy are known for $pr\mathcal{BPP}$ but not for $\mathcal{BPP}$.

APEP. *(In fact,* APEP *is the problem of estimating the acceptance probability of a given Boolean circuit, mentioned in Proposition 2.)*

2. **(Search-to-decision.)** *Solving any $\mathcal{PPT}$-search problem is reducible to pr$\mathcal{BPP}$ in deterministic polynomial time.*

3. **(A form of error-reduction.)** *For any $\mathcal{PPT}$-search problem and any $c \in \mathbb{N}$, there is a ppt algorithm that solves the problem with success probability $1 - 2^{-n^c}$. (And this statement remains true if we change the success probability in Item (2) of Definition 1 to be $1/\mathrm{poly}(n)$ rather than $2/3$.)*

4. **(Hierarchy.)** *For every polynomial $T(n) \geq n$ there is a $\mathcal{PPT}$-search problem that is solvable in time $\tilde{O}(T)$ but not in time T.*

The proof of Theorem 3 appears in Section B. In addition, as mentioned in the introduction, the class of $\mathcal{PPT}$-search problems (when solving is defined as weakly solving) is computationally equivalent to the optimization-based class of problems that we propose; see Section 3.2 for details.

Remark 4. *Some alternative "promise-problem" formulations of the class of search problems solvable in ppt satisfy a subset of the properties stated in Theorem 3, but these alternatives (provably) do not satisfy all of the properties. For further discussion see Appendix A, as well as the discussion of "Type 2" and "Type 3" problems in [Gol24, Sections 2.1 and 2.2].*

The main shortcomings of our suggested definition. When defining solving a $\mathcal{PPT}$-search problem as weakly solving, the rationale for insisting on the existence of an algorithm strongly solving the problem in Definition 1 is not clear. One may think of the resulting complexity class as a subclass of the broader class "all problems weakly solvable in ppt", but it is not clear what is the a-priori *conceptual* reason to focus on this particular subclass (i.e., aside from the a-posteriori fact that it turns out to be a useful class, as demonstrated in Proposition 2, Theorem 3 and Section 3.2).

Another issue is that the error-reduction in Item (3) of Theorem 3 is not what we would ideally want: It starts with an algorithm that strongly solves the problem with large error, and yields an algorithm that *weakly* solves the problem with small error.

3 Optimization: An alternative that avoids these shortcomings

As mentioned above, we suggest an alternative definition that avoids the shortcomings mentioned in Section 2.3, and that is focused on a notion of optimization.

3.1 The suggested definition of $\mathcal{PPT}$-optimization

To motivate the new definition, fix an instance x to a search problem, and consider the tripartition of potential solutions from Section 2.3 to "good" solutions $y \in \Pi^{\mathsf{Y}}(x)$, "adequate" solutions $y \notin (\Pi^{\mathsf{Y}}(x) \cup \Pi^{\mathsf{N}}(x))$, and "bad" solutions $y \in \Pi^{\mathsf{N}}(x)$. Let us

think of each y as assigned a quality value $q(x,y) \in \{1, 1/2, 0\}$, where 1 and $1/2$ and 0 represent the three foregoing cases, respectively. Instead of insisting on a discrete tripartition, we will consider a smoother assignment of quality values, where each (x,y) is assigned a real value $q(x,y) \in [0,1]$. And instead of requiring an algorithm that solves the promise-problem represented by the tripartition, we will require an algorithm that (approximately) computes the quality $q(x,y)$ of each (x,y).

This naturally yields a notion of an **optimization problem**, where each potential solution is assigned a quality value, and we are interested in finding the solution of **best quality**. Since we are interested in problems where quality can only be approximated (rather than computed exactly), we will also allow outputting a solution of *approximately* best quality. This yields the following definition:

Definition 5 ($\mathcal{PPT}$-optimization)**.** *Let $P \subseteq \{0,1\}^*$ (representing a promise on instances), and let $q\colon P \times \{0,1\}^* \to [0,1]$. We say that (P,q) is a $\mathcal{PPT}$-optimization problem if the following holds:*

1. *There is a ppt algorithm V that gets input $(x,y,1^t)$ where $x \in P$, and satisfies $\Pr[|V(x,y,1^t) - q(x,y)| \leq 1/t] \geq 2/3$.*
2. *There is a ppt algorithm F that gets input $(x,1^t)$ where $x \in P$, and satisfies $\Pr[q(x,F(x,1^t)) \geq q(x) - 1/t] \geq 2/3$, where $q(x) = \max_y \{q(x,y)\}$.*

We now explicitly define the notion of **solving a $\mathcal{PPT}$-optimization problem:** Given input $(x,1^t)$, one is required to output a solution of quality at least $q(x) - 1/t$. Indeed, the requirement in Definition 5 is that there exists a ppt algorithm solving the problem, and there is now no inconsistency between the notion of solving the problem and the requirements of the algorithm posed in the definition.

Thus, the formulation of $\mathcal{PPT}$-optimization is more consistent than the formulation of $\mathcal{PPT}$-search, and its meaning is clearer.

Remark 6. *The formulation in Definition 5 is almost identical to the main one suggested by Goldreich [Gol24, Definition 2.4], the only difference being that we allow the promise P on the instances. This difference seems meaningful when the promise is not efficiently recognizable.*[5] *Also, for clarity, one may choose to enforce an additional condition that all solutions are of length* $\mathrm{poly}(|x|)$*, although this does not seem technically necessary to us.*

3.2 Equivalence of $\mathcal{PPT}$-search and $\mathcal{PPT}$-optimization

We show that the class of problems given in Definition 5 is computationally equivalent to the class given by our interpretation of Definition 1, in the following sense: Weakly solving any $\mathcal{PPT}$-search problem is reducible in deterministic polynomial

[5] To see this, consider the naive reduction of any $\mathcal{PPT}$-optimization problem (P,q) with a promise to total $\mathcal{PPT}$-optimization problem q', obtained by defining q' trivially on inputs outside the promise. The issue is that q' needs to be efficiently approximable, and this seemingly requires that P will be efficiently recognizable. (For example, if we try to define q' using the hypothesized ppt approximation algorithm V for q, it may be that V "misbehaves" outside the promise and does not allow for defining q'.)

time to solving a $\mathcal{PPT}$-optimization problem, and vice versa. This can be viewed as analogous to the computational equivalence between search and optimization in the context of $\mathcal{NP}$ problems (see, e.g., [Gol08, Section 2.2.2], [Aar16, Section 2.2.1]).

Theorem 7 ($\mathcal{PPT}$-search and $\mathcal{PPT}$-optimization are computationally equivalent)**.** *The following two statements are true:*

1. *For any $\mathcal{PPT}$-search problem Π there is a $\mathcal{PPT}$-optimization problem (P, q) such that weakly solving Π reduces in deterministic polynomial time to solving (P, q).*

2. *For any $\mathcal{PPT}$-optimization problem (P, q) there is a $\mathcal{PPT}$-search problem Π such that solving q reduces in deterministic polynomial time to weakly solving Π.*

Moreover, in both items, the reduction is extremely efficient (i.e., more than just "deterministic polynomial time"): In one direction the reduction maps instance x to instance $(x, 1^4)$, and in the other direction the reduction maps instance $(x, 1^t)$ to $(x, 1^{2t})$.

Before turning to the proof of Theorem 7, let us point out some implications of the computational equivalence:

- The class $\mathcal{PPT}$-optimization captures a set of problems with essentially the same complexity as the set of problems captured by $\mathcal{PPT}$-search, so we did not lose or overshoot when moving from search to optimization.

- Moreover, examining the proof below, there is a close correspondence between each problem in $\mathcal{PPT}$-search and its reduction target in $\mathcal{PPT}$-optimization, and vice versa. Thus, the two classes seem to capture very similar sets of problems.

- When trying to bound the complexity of search problems solvable in ppt, we can equivalently use either formulation (i.e., $\mathcal{PPT}$-search or $\mathcal{PPT}$-optimization), according to what is technically expedient.

In addition, the equivalence lends further support to our suggestion to define solving $\mathcal{PPT}$-search problems as weakly solving in Section 2 (since this definition yields a class that is computationally equivalent to $\mathcal{PPT}$-optimization).

Proof of Theorem 7. For Item (1), let $\Pi = (\Pi^{\mathsf{Y}}, \Pi^{\mathsf{N}})$ and let V, F be the algorithms for Π from Definition 1. We define $P \subseteq \{0,1\}^*$ as the set of instances x such that $\Pi^{\mathsf{Y}}(x) \neq \emptyset$, and for every $x \in P$ and $y \in \{0,1\}^*$, we define $q(x, y) = \min\{\Pr[V(x, y) = 1], 2/3\}$. Indeed, (P, q) is a $\mathcal{PPT}$-optimization problem, since we can approximate q in ppt (i.e., by simulating V), and since given $x \in P$ we can find in polynomial time a solution y of quality $q(x)$, whp (i.e., since $\Pr_{z \leftarrow F(x)}[q(z) = 2/3] \geq 2/3$).[6]

To reduce (weakly) solving Π to solving (P, q), we map an instance x for Π to an instance $(x, 1^4)$ for (P, q). For any algorithm F_q that solves (P, q), we have that $q(x, F_q(x, 1^4)) \geq q(x) - 1/4$. Hence, for every x such that $\Pi^{\mathsf{Y}}(x) \neq \emptyset$ (meaning that

[6] Indeed, note that the search algorithm for q gets input $(x, 1^t)$ and finds a solution of optimal quality $q(x)$, rather than of near-optimal quality $q(x) - 1/t$.

$q(x) = 2/3$), the algorithm $F_q(x, 4)$ outputs z such that $\Pr[V(x,z)] \geq q(x,z) \geq 2/3 - 1/4 > 1/2$, which implies that $z \notin \Pi^{\mathsf{N}}(x)$.

For Item (1), let (P,q) be a $\mathcal{PPT}$-optimization problem, and let V, F be the algorithms from Definition 5. We define a promise-problem $\Pi = (\Pi^{\mathsf{Y}}, \Pi^{\mathsf{N}})$ in which instances are of the form $(x, 1^t)$, and $\Pi^{\mathsf{Y}}(x, 1^t) = \{y : q(x,y) \geq q(x) - 1/t\}$ and $\Pi^{\mathsf{N}}(x, 1^t) = \{y : q(x,y) \leq q(x) - 2/t\}$.

Let us first argue that Π is a $\mathcal{PPT}$-search problem. We can solve the decision problem of Π by using both F and V: We first use $F(x, 1^{t^2})$ to approximate $q(x)$ with accuracy $1/t^2$, then use $V(x, y, 1^{t^2})$ to approximate $q(x,y)$ with accuracy $1/t^2$, and we accept (x,y) iff $\tilde{q}(x,y) \geq \tilde{q}(x) - 3/2t$, where $\tilde{q}$ denotes our approximation of q in both cases. Also, we can solve the search problem of Π using F (since $F(x, 1^t)$ outputs z such that whp $q(x,z) \geq q(x) - 1/t$, meaning that $z \in \Pi^{\mathsf{Y}}(x, 1^t)$).

The reduction of (P,q) to Π maps an instance $(x, 1^t)$ for (P,q) to an instance $(x, 1^{2t})$ for Π. Any algorithm that (weakly) solves Π finds $y \notin \Pi^{\mathsf{N}}(x, 1^{2t})$, meaning that $q(x,y) > q(x) - 2/(2t) = q(x) - 1/t$, as required for solving (P,q). ■

3.3 Properties of $\mathcal{PPT}$-optimization

The class $\mathcal{PPT}$-optimization captures natural problems that we are interested in. First, the three problems mentioned in Proposition 2 have direct analogues in $\mathcal{PPT}$-optimization.[7] And secondly, any probabilistic polynomial-time FPTAS (i.e., a fully polynomial-time approximation scheme) for a function taking values in $[0,1]$ yields a corresponding $\mathcal{PPT}$-optimization problem; see [Gol24, Theorem 2.6].

In addition, since any $\mathcal{PPT}$-optimization problem is reducible to a $\mathcal{PPT}$-search problem and vice versa, the class of $\mathcal{PPT}$-optimization problems enjoys valuable technical properties analogous to the ones mentioned in Theorem 3. That is, for each property asserted in Theorem 3, an analogous property for $\mathcal{PPT}$-optimization can be established by reductions to $\mathcal{PPT}$-search and back.[8] However, it may be more informative to present the foregoing properties of $\mathcal{PPT}$-optimization explicitly and provide direct proofs. Let us do so now:

Theorem 8 (technical properties of $\mathcal{PPT}$-optimization)**.** *The class of $\mathcal{PPT}$-optimization problems satisfies the following properties:*

[7] For estimating the acceptance probability of a circuit, we present a problem called OAPEP that is complete for $\mathcal{PPT}$-optimization; see Theorem 8. For finding fooling sets for a circuit, we can define solutions as sets, where the quality of a set S with respect to a circuit $C\colon \{0,1\}^n \to \{0,1\}$ is $|\Pr_{s\in S}[C(s) = 1] - \Pr_{r\in\{0,1\}^n}[C(r) = 1]|$. For the search version of PIT, given circuit C, we define the quality of x as the binary indicator of whether $C(x) \neq 0$ (to be more precise, a solution for PIT also includes a representation of $\mathbb{K} \supseteq \mathbb{F}$ where $x \in \mathbb{K}^n$; see the proof of Proposition 12).

[8] Specifically, the class of $\mathcal{PPT}$-optimization problems has a complete problem (obtained by reducing to $\mathcal{PPT}$-search, reducing to APEP, and reducing APEP back to $\mathcal{PPT}$-search); a search-to-decision reduction (obtained by reducing to $\mathcal{PPT}$-search and then reducing to $pr\mathcal{BPP}$); error-reduction (obtained by reducing to $\mathcal{PPT}$-search and using the small-error algorithm for the latter); and a time hierarchy (again obtained by the two-way reductions, which are time-efficient).

1. **(Complete problem.)** *There is a total $\mathcal{PPT}$-optimization problem denoted* OAPEP *such that any $\mathcal{PPT}$-optimization problem reduces to* OAPEP *in deterministic polynomial time. (Intuitively,* OAPEP *asks to find a near-optimal approximation for the acceptance probability of a given circuit.)*

2. **(Search-to-decision.)** *Solving $\mathcal{PPT}$-optimization problem is reducible to $pr\mathcal{BPP}$ in deterministic polynomial time.*

3. **(Error-reduction.)** *For any $\mathcal{PPT}$-optimization problem and any $c \in \mathbb{N}$, there is a ppt algorithm that solves the problem with success probability $1 - 2^{-n^c}$. (This remains true if we change the success probability in Item (2) of Definition 5 to be $1/\mathrm{poly}(n)$.)*

4. **(Hierarchy.)** *For every polynomial $T(n) \geq n$, there is a $\mathcal{PPT}$-optimization problem that is solvable in time $\tilde{O}(T)$ but not in time T.*

Proof. For proofs of Items (2) and (3), see [Gol24, Section 4.1].

For Item (1), consider the following definition of OAPEP. The instances are Boolean circuits $C\colon \{0,1\}^n \to \{0,1\}$, and for any solution $v \in [0,1]$, we define its quality to be $q(C,v) = |\Pr_r[C(r) = 1] - v|$. Note that OAPEP is indeed a $\mathcal{PPT}$-optimization problem, since given $(C, v, 1^t)$ we can estimate $q(C,v)$ up to accuracy $1/t$ in time $\mathrm{poly}(|C|, |v|, t)$; and since given $(C, 1^t)$ we can output v such that $|\Pr_r[C(r) = 1] - v| \leq 1/O(t)$ in time $\mathrm{poly}(t, |C|)$, by naive sampling.

To see that any $\mathcal{PPT}$-optimization problem reduces to OAPEP in deterministic polynomial time, observe that by Item (2), any $\mathcal{PPT}$-optimization problem reduces to a decision problem $pr\mathcal{BPP}$. Since CAPP is complete for $pr\mathcal{BPP}$,[9] the latter decision problem reduces to CAPP. Finally, CAPP for a circuit C reduces to OAPEP for $(C, 1/7)$ (i.e., since solving OAPEP on input $(C, 1/7)$ allows obtaining an approximation of C's acceptance probability up to accuracy smaller than $1/6$, which suffices for CAPP).

For Item (4), we formulate any decision problem in $pr\mathcal{BPP}$ as a $\mathcal{PPT}$-optimization problem, and rely on the known hierarchy result for $pr\mathcal{BPP}$ (see, e.g., [Bar02]). Specifically, we will consider $\mathcal{PPT}$-optimization problems wherein solutions are bits, and the quality of each bit is binary. Letting $\Pi = (\Pi^{\mathsf{Y}}, \Pi^{\mathsf{N}}) \in pr\mathcal{BPP}$, define a $\mathcal{PPT}$-optimization problem (P, q) by $P = \Pi^{\mathsf{Y}} \cup \Pi^{\mathsf{N}}$, and for any $x \in \Pi^{\mathsf{Y}}$ define $q(x,b) = b$, and for any $x \in \Pi^{\mathsf{N}}$ define $q(x,b) = 1 - b$ (where $b \in \{0,1\}$). The fact that (P,q) is a $\mathcal{PPT}$-optimization problem follows since $\Pi \in pr\mathcal{BPP}$.[10]

Now, let $\Pi \in pr\mathcal{BPTIME}[\tilde{O}(T)] \setminus pr\mathcal{BPTIME}[O(T)]$, for some polynomial T, and let (P,q) be the $\mathcal{PPT}$-optimization problem described above. Since $\Pi = pr\mathcal{BPTIME}[\tilde{O}(T)]$ it follows that q is solvable in time $\tilde{O}(T)$ (i.e., given $(x, 1^t)$ where $x \in P$ we can compute the exact value of $q(x,0)$ and of $q(x,1)$ in time $\tilde{O}(T)$). Assuming towards a contradiction that q is solvable by an algorithm F running in time T, we

[9] Recall that CAPP is the promise-problem of distinguishing between circuits with acceptance probability at least $2/3$ and circuits with acceptance probability at most $1/3$.

[10] Specifically, using the decision algorithm for Π, for any $x \in P$ we can compute the quality of $(x,0)$ and of $(x,1)$, which also means that we can find $b \in \{0,1\}$ with the best quality.

can also solve Π in time $O(T)$; specifically, given $x \in \Pi^{\mathsf{Y}} \cup \Pi^{\mathsf{N}}$, run $F(x, 1^4)$ to find b such that $q(x, b)$ is maximal and output b. ■

Appendix A Critiques of some other alternatives

In this appendix we discuss three alternative definitions for the class of search problems solvable in ppt that are based on promise-problems.

In all the definitions below, a search problem is represented as a promise-problem $\Pi = (\Pi^{\mathsf{Y}}, \Pi^{\mathsf{N}})$ over pairs (i.e., Π^{Y} and Π^{N} are disjoint relations as in Definition 1), and we assume that a ppt verification algorithm can solve the decision problem of Π. Thus, the main difference is in the search algorithm that we require for a problem to be in the class, and in what we define solving a problem to mean.

A.1 The first alternative: Solving means strongly solving

Starting from Definition 1, we can define solving a problem as *strongly solving*, in which case there is no inconsistency between the requirements from the algorithm in Item (2) of Definition 1 and the requirements for solving the problem.

The main problem in this formulation is that the requirement from a search algorithm (i.e., strongly solving) is stricter than the requirement from the verification algorithm V (i.e., limbo solutions can be accepted). To see why this is a serious problem, recall that we are looking for a class in which solutions can be found efficiently and verified efficiently, where "efficiently" means probabilistic polynomial-time. In particular, we expect all search problems in the class to be solvable in deterministic exponential time, by brute-force enumeration over all candidate solutions. However, for some problems satisfying the requirements of Definition 1, no deterministic algorithm (of any running time) can strongly solve the problem. That is:

Claim 9. *There is a problem Π satisfying Definition 1 such that there does not exist a deterministic uniform algorithm (of any running time) strongly solving Π.*

Proof. The instances in Π are unary. For every $n \in \mathbb{N}$, we define $\Pi^{\mathsf{Y}}(1^n)$ as the set of n-bit strings with Kolmogorov complexity at least $n - O(1)$, and we define $\Pi^{\mathsf{N}}(1^n) = \{0^n\}$. Note that Π satisfies Definition 1, because there is a polynomial-time algorithm V that solves the decision problem of Π (i.e., the algorithm accepts $(1^n, y)$ iff $y \in \{0,1\}^n \setminus \{0^n\}$) and there is a ppt algorithm that gets 1^n and outputs $y \in \Pi^{\mathsf{Y}}(1^n)$, whp (i.e., by sampling $y \in \{0,1\}^n$ at random). However, no deterministic algorithm can map 1^n to an n-bit string with Kolmogorov complexity $n - O(1)$. ■

The reason that Claim 9 holds is that the verification algorithm V accepts any string in $\{0,1\}^* \setminus \{0^n\}$, whereas the search algorithm is required to find a solution in the smaller set $\Pi^{\mathsf{Y}}(1^n)$. As one corollary of Claim 9, the formulation in which solving means strongly solving does not support a deterministic search-to-decision reduction (i.e., a deterministic reduction of strongly solving the problem to $pr\mathcal{BPP} \subseteq pr\mathcal{EXP}$).

A.2 The second alternative: Only require weakly solving to be in the class

Alternatively, we may modify Item (2) of Definition 1, and only require a ppt algorithm that *weakly solves* the problem in order to say that a problem is in the class. We now define solving the problem as weakly solving, and there is no inconsistency between the notion of solving and the requirements in (the modified) Item (2). Also, unlike the alternative in Section A.1, the requirement from a search algorithm is not stricter than the requirement from the verification algorithm V.

The main problem is that this formulation is overly broad, capturing infeasible problems that we do not want our notion to capture. In fact, for *any* search problem R in which solutions are verifiable (e.g., the problem of finding a satisfying assignment for a given Boolean circuit), a padded version of R belongs to the resulting class:

Claim 10. *Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ such that $R \in \mathcal{BPP}$, and let $\Pi = (\Pi^{\mathsf{Y}}, \Pi^{\mathsf{N}})$ such that $\Pi^{\mathsf{Y}} = \{(x, y1) : (x,y) \in R\}$, and $\Pi^{\mathsf{N}} = \{(x, y1) : (x,y) \notin R\}$, and limbo solutions are strings ending in zero. Then, the search problem Π satisfies the formulation above.*

Proof. The claim follows since a search algorithm can output a string ending with zero (which is a limbo solution). ■

In addition, this formulation does not support an efficient (deterministic) search-to-decision reduction. Specifically:

Claim 11. *Assuming that $\mathcal{NP} \not\subseteq \mathcal{BPP}$, there is a problem Π satisfying the requirements of the formulation above such that there is no deterministic polynomial-time reduction of solving Π to $pr\mathcal{BPP}$.*

Proof. Consider the problem $\Pi = (\Pi^{\mathsf{Y}}, \Pi^{\mathsf{N}})$ in which instances are Boolean circuits $C\colon \{0,1\}^n \to \{0,1\}$, and for every C we define

$$\Pi^{\mathsf{Y}}(C) = \left\{(x,y) \in \{0,1\}^n \times \{0,1\}^{|C|^2} : C(x) = 1\right\}$$
$$\Pi^{\mathsf{N}}(C) = \left\{(x,y) \in \{0,1\}^n \times \{0,1\}^{|C|^2} : C(x) = 0 \wedge \mathsf{K}(y) \leq |C|^2 - O(1)\right\} ,$$

where $\mathsf{K}(y)$ is the Kolmogorov complexity of y. Observe that Π satisfies the requirements of the formulation above, since there is an algorithm solving the decision problem Π (i.e., it accepts $(C,(x,y))$ iff $C(x) = 1$) and there is a ppt algorithm that gets C and finds solutions not in $\Pi^{\mathsf{N}}(C)$ (i.e., it chooses a uniformly random string y, which has high Kolmogorov complexity whp, and outputs $(1^n, y)$).

Assume towards a contradiction that there is a deterministic polynomial-time oracle machine A such that $A^{pr\mathcal{BPP}}$ weakly solves Π. Observe that there are at most finitely many inputs $C\colon \{0,1\}^n \to \{0,1\}$ such that $A^{pr\mathcal{BPP}}(C)$ prints (x,y) with $\mathsf{K}(y) > |C|^2 - O(1)$; this is the case since $A^{pr\mathcal{BPP}}$ can be simulated by a deterministic exponential-time algorithm A', and since for every such C we have that $\mathsf{K}(y) \leq |A'| + |C| + O(1) = O(|C|)$. Hence, for all but finitely many inputs C the algorithm $A^{pr\mathcal{BPP}}(C)$ prints (x,y) such that $C(x) = 1$ (since $(x,y) \notin \Pi^{\mathsf{N}}(C)$ and $\mathsf{K}(y) \leq |C|^2 - O(1)$). This means that $A^{pr\mathcal{BPP}}$ solves the $\mathcal{NP}$-complete problem CircuitSAT, and hence $\mathcal{NP} \subseteq \mathcal{BPP}$. ■

A.3 The third alternative: Reductions to $pr\mathcal{BPP}$

Starting from Definition 1, let us consider a stricter efficieny requirement on the search algorithm in Item (2): Instead of requiring a ppt algorithm that finds solutions in $\Pi^{\mathsf{Y}}(x)$, we require a deterministic polynomial-time reduction of finding solutions in $\Pi^{\mathsf{Y}}(x)$ to $pr\mathcal{BPP}$ (i.e., there is a deterministic polynomial-time machine making queries to $pr\mathcal{BPP}$ that gets x for which $\Pi^{\mathsf{Y}}(x) \neq \emptyset$ and outputs $y \in \Pi^{\mathsf{Y}}(x)$).

The advantage in this formulation is that search-to-decision and error-reduction are built-in to the class from the definition, regardless of whether we define solving a problem as strongly solving or as weakly solving. The main disadvantage is that there is no evidence that this formulation captures the intuitive notion that we are interested in. In fact, the set of problems satisfying this formulation is a *strict subset* of $\mathcal{PPT}$-search problems as in Definition 1 (where strictness is due to Claim 9).

Nevertheless, this set of search problems seems to be of independent interest. An intuitive meaning for this set is "the subset of $\mathcal{PPT}$-search problems that can be strongly solved by deterministic polynomial-time reductions to $pr\mathcal{BPP}$". We therefore suggest the following natural open problem:

Open Problem 1. *Find interesting characterizations of the subset of $\mathcal{PPT}$-search problems that can be strongly solved by deterministic polynomial-time reductions to $pr\mathcal{BPP}$.*

As shown in [Gol11, Discussion after Theorem 3.5], any $\mathcal{PPT}$-search problem that has a *companion problem* in $\mathcal{PPT}$-search belongs to the foregoing subset.

Appendix B Deferred proofs

Finally, we include formal statements of Proposition 2 and of Theorem 3 and proofs for them. Let us begin with Proposition 2.

Proposition 12 (Proposition 2, stated formally)**.** *The following problems are all $\mathcal{PPT}$-search problems:*

1. APEP *(Acceptance Probability Estimation Problem [PDV21]):*
   - *The instances are pairs $(C, 1^t)$ where $C\colon \{0,1\}^n \to \{0,1\}$ is a Boolean circuit and $t \in \mathbb{N}$ is an integer. Solutions are real numbers $v \in [0,1]$.*
   - *"Yes" solutions for $(C, 1^t)$ satisfy $|\Pr_r[C(r) = 1] - v| \leq 1/t$.*
   - *"No" solutions for $(C, 1^t)$ satisfy $|\Pr_r[C(r) = 1] - v| \geq 2/t$.*
2. FoolingSet*:*
   - *The instances are pairs $(C, 1^t)$ where $C\colon \{0,1\}^n \to \{0,1\}$ is a Boolean circuit and $t \in \mathbb{N}$ is an integer. Solutions are sets $S \subseteq \{0,1\}^n$.*
   - *"Yes" solutions for $(C, 1^t)$ satisfy $|\Pr_r[C(r) = 1] - \Pr_s[C(s) = 1]| \leq 1/t$.*
   - *"No" solutions for $(C, 1^t)$ satisfy $|\Pr_r[C(r) = 1] - \Pr_s[C(s) = 1]| \geq 2/t$.*

3. search-PIT:

   - *The instances are pairs (ρ, C) where ρ is a representation of a field $\mathbb{F}$ (i.e., a prime number and possibly an irreducible polynomial), and C is an n-variate arithmetic circuit over $\mathbb{F}$. Solutions are pairs (ρ', x) where ρ' is a representation of a field $\mathbb{K} \supseteq \mathbb{F}$ (it may be that $\mathbb{K} = \mathbb{F}$) and $x \in \mathbb{K}^n$.*
   - *"Yes" solutions for (ρ, C) satisfy $C(x) \neq 0$.*
   - *"No" solutions for (ρ, C) satisfy $C(x) = 0$.*

For all problems above, if $pr\mathcal{BPP} = pr\mathcal{P}$ then there is a deterministic polynomial-time algorithm that (weakly) solves the problem.

Proof. The fact that APEP is in $\mathcal{PPT}$-search follows using the natural algorithm that estimates the acceptance probability of a given circuit up to accuracy (say) $1/t^2$ (i.e., we use this algorithm both for verification V and for finding F).

For FoolingSet, the verification algorithm V gets $(C, S, 1^t)$ and distinguishes between the "yes" and "no" cases by estimating $\Pr_r[C(r) = 1]$ up to accuracy (say) $1/t^2$, using naive sampling; and the search algorithm F outputs a random sample $S \subseteq \{0,1\}^n$ of size $O(t^2)$.

As for search-PIT, verifying a solution is straightforward (i.e., by evaluating $C(x)$). To find solutions, let $d \leq 2^{|C|}$ be the degree of the given circuit C, and note that: (1) We can find a field $\mathbb{K} \supseteq \mathbb{F}$ of size at least $3d$ in probabilistic time $\text{polylog}(d) \leq \text{poly}(|C|)$, with high probability (see, e.g., [Sho90, Theorem 5.1]); (2) By the DeMillo–Lipton–Schwartz–Zippel lemma, a random element $x \in \mathbb{K}^n$ is a solution, with probability at least $1 - d/|\mathbb{K}|$ (see, e.g., [Gol08, Lemma 6.8], [AB09, Lemma 7.5]).

The fact that $pr\mathcal{BPP} = pr\mathcal{P}$ implies deterministic polynomial-time algorithms for these problems follows from Item (2) of Theorem 3 (which will be proved below). ■

Let us restate Theorem 3 formally and prove it:

Theorem 13 (technical properties of our interpretation of $\mathcal{PPT}$-search; Theorem 3, restated)**.** *Consider the class of $\mathcal{PPT}$-search problem wherein solving the problem means* **weakly solving**. *Then, the following properties hold:*

1. **(Complete problem.)** *There is a total $\mathcal{PPT}$-search problem* APEP *such that solving any $\mathcal{PPT}$-search reduces in deterministic polynomial time to solving* APEP.

2. **(Search-to-decision.)** *Solving any $\mathcal{PPT}$-search problem is reducible to $pr\mathcal{BPP}$ in deterministic polynomial time.*

3. **(Some form of error-reduction.)** *For any $\mathcal{PPT}$-search problem and any $c \in \mathbb{N}$, there is a ppt algorithm that solves the problem with success probability $1 - 2^{-n^c}$. (And this statement remains true if we change the success probability in Item (2) of Definition 1 to be $1/\text{poly}(n)$ rather than $2/3$.)*

4. **(Hierarchy.)** *For every polynomial $T(n) \geq n$ there is a $\mathcal{PPT}$-search problem that is solvable in time $\tilde{O}(T)$ but not in time T.*

Proof. Item (2) was proved in [Gol11, Theorem 3.5]. (The original statement did not use this terminology, but the statement immediately implies Item (2).)

For Item (1), to see that APEP is complete for $\mathcal{PPT}$-search, we first reduce any problem in $\mathcal{PPT}$-search to $pr\mathcal{BPP}$ (i.e., using the search-to-decision reduction of Item (2)), and then reduce $pr\mathcal{BPP}$ to APEP. For the latter reduction, recall that $pr\mathcal{BPP}$ reduces to CAPP, and note that CAPP reduces to APEP in the obvious way.

For Item (3), given x we invoke $F(x)$ multiple times and test each outcome using the algorithm for Π, trying to find an outcome that does not belong to $\Pi^{\mathsf{N}}(x)$. Specifically, assume that $F(x)$ succeeds in finding $y \in \Pi^{\mathsf{Y}}(x)$ with probability $1/\mathrm{poly}(n)$. First, we increase the success probability of the verification algorithm V to $1 - 2^{-\mathrm{poly}(n)}$ (using naive error-reduction). Running F for $\mathrm{poly}(n)$ times, with probability $1 - 2^{-\mathrm{poly}(n)}$ the following two statements hold: There is at least one outcome in $\Pi^{\mathsf{Y}}(x)$, and V rejects all outcomes that are in $\Pi^{\mathsf{N}}(x)$.

For Item (4), similarly to the proof of Theorem 8, we formulate any decision problem in $pr\mathcal{BPP}$ as a $\mathcal{PPT}$-search problem, and rely on the time hierarchy for $pr\mathcal{BPP}$. Specifically, letting $\Pi_0 = (\Pi_0^{\mathsf{Y}}, \Pi_0^{\mathsf{N}}) \in pr\mathcal{BPTIME}[\tilde{O}(T)] \setminus pr\mathcal{BPTIME}[O(T)]$, for any $x \in \Pi_0^{\mathsf{Y}}$ define $\Pi^{\mathsf{Y}}(x) = \{1\}$ and $\Pi^{\mathsf{N}}(x) = \{0\}$, and for any $x \in \Pi_0^{\mathsf{N}}$ define $\Pi^{\mathsf{Y}}(x) = \{0\}$ and $\Pi^{\mathsf{N}}(x) = \{1\}$. Then $\Pi = (\Pi^{\mathsf{Y}}, \Pi^{\mathsf{N}})$ is in $\mathcal{PPT}$-search, and Π is solvable in time $\tilde{O}(T)$, and an algorithm F that solves Π in time T yields an algorithm that decides Π_0 in time $O(T)$. ■

References

[Aar16] Scott Aaronson. "$P \stackrel{?}{=} NP$". In: *Open Problems in Mathematics*. Ed. by John Forbes Nash Jr. and Michael Th. Rassias. Springer International Publishing, 2016, pp. 1–122.

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: A modern approach*. Cambridge University Press, Cambridge, 2009.

[Bar02] Boaz Barak. "A probabilistic-time hierarchy theorem for "slightly non-uniform" algorithms". In: *Proc. 6th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*. Vol. 2483. 2002, pp. 194–208.

[Gol08] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. New York, NY, USA: Cambridge University Press, 2008.

[Gol11] Oded Goldreich. "In a World of P=BPP". In: *Studies in Complexity and Cryptography. Miscellanea on the Interplay Randomness and Computation*. 2011, pp. 191–232.

[Gol24] Oded Goldreich. *On Defining PPT-Search Problems*. 2024.

[PDV21] A. Pavan Peter Dixon and N.V. Vinodchandran. “Promise Problems Meet Pseudodeterminism”. In: *Electronic Colloquium on Computational Complexity: ECCC* 28 (2021), p. 043.

[Sho90] Victor Shoup. “New algorithms for finding irreducible polynomials over finite fields”. In: *Mathematics of Computation* 54.189 (1990), pp. 435–447.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il