Journal of Artificial Intelligence Research 45 (2012) 685-729 Submitted 08/12; published 12/12

The Time Complexity of A^* with Approximate Heuristics on Multiple-Solution Search Spaces

Hang Dinh HTDINH@IUSB.EDU
Department of Computer & Information Sciences
Indiana University South Bend
1700 Mishawaka Ave. P.O. Box 7111
South Bend, IN 46634 USA

Hieu Dinh HIEU.DINH@MATHWORKS.COM
MathWorks
3 Apple Hill Drive
Natick, MA 01760-2098 USA

Laurent Michel LDM@ENGR.UCONN.EDU
Alexander Russell ACR@CSE.UCONN.EDU
Department of Computer Science & Engineering
University of Connecticut
371 Fairfield Way, Unit 2155
Storrs, CT 06269-2155 USA

Abstract

We study the behavior of the A^* search algorithm when coupled with a heuristic h satisfying $(1-\epsilon_1)h^* \leq h \leq (1+\epsilon_2)h^*$, where $\epsilon_1, \epsilon_2 \in [0,1)$ are small constants and h^* denotes the optimal cost to a solution. We prove a rigorous, general upper bound on the time complexity of A^* search on trees that depends on both the accuracy of the heuristic and the distribution of solutions. Our upper bound is essentially tight in the worst case; in fact, we show nearly matching lower bounds that are attained even by non-adversarially chosen solution sets induced by a simple stochastic model. A consequence of our rigorous results is that the effective branching factor of the search will be reduced as long as $\epsilon_1 + \epsilon_2 < 1$ and the number of near-optimal solutions in the search tree is not too large. We go on to provide an upper bound for A^* search on graphs and in this context establish a bound on running time determined by the spectrum of the graph.

We then experimentally explore to what extent our rigorous upper bounds predict the behavior of A^* in some natural, combinatorially-rich search spaces. We begin by applying A^* to solve the knapsack problem with near-accurate admissible heuristics constructed from an efficient approximation algorithm for this problem. We additionally apply our analysis of A^* search for the partial Latin square problem, where we can provide quite exact analytic bounds on the number of near-optimal solutions. These results demonstrate a dramatic reduction in effective branching factor of A^* when coupled with near-accurate heuristics in search spaces with suitably sparse solution sets.

1. Introduction

The classical A^* search procedure (Hart, Nilson, & Raphael, 1968) is a method for bringing heuristic information to bear on a natural class of search problems. One of A^*'s celebrated features is that when coupled with an *admissible* heuristic function, that is, one that always returns a lower bound on the distance to a solution, A^* is guaranteed to find an optimal solution. While the worst-case behavior of A^* (even with an admissible heuristic function) is no better than that of, say, breadth-first search, both practice and intuition suggest that availability of an accurate heuristic should decrease the running time. Indeed, methods for computing accurate admissible heuristic functions for various search problems have been presented in the literature (see, e.g., Felner, Korf, & Hanan, 2004). In this article, we investigate the effect of such accuracy on the running time of A^* search;

specifically, we focus on rigorous estimates for the running time of A^* when coupled with accurate heuristics.

The initial notion of *accuracy* we adopt is motivated by the standard framework of approximation algorithms: if $f(\cdot)$ is a hard combinatorial optimization problem (e.g., the permanent of a matrix, the value of an Euclidean traveling salesman problem, etc.), an algorithm $\mathcal{A}$ is an efficient *ϵ-approximation* to f if $\mathcal{A}$ runs in polynomial time and $(1-\epsilon)f(x) \leq \mathcal{A}(x) \leq (1+\epsilon)f(x)$, for all inputs x, where $f(x)$ is the optimal solution cost for input x and $\mathcal{A}(x)$ is the solution cost returned by algorithm $\mathcal{A}$ on input x. The approximation algorithms community has developed efficient approximation algorithms for a wide swath of NP-hard combinatorial optimization problems and, in some cases, provided dramatic lower bounds asserting that various problems cannot be approximated beyond certain thresholds (see Vazirani, 2001; Hochbaum, 1996, for surveys of this literature). Considering the great multiplicity of problems that have been successfully addressed in this way (including problems believed to be far outside of NP, like matrix permanent), it is natural to study the behavior of A^* when coupled with a heuristic function possessing such properties. Indeed, in some interesting cases (e.g., Euclidean travelling salesman, matrix permanent, knapsack), hard combinatorial problems can be approximated in polynomial time to within *any fixed constant* $\epsilon > 0$; in these cases, the polynomial depends on the constant ϵ. We remark, also, that many celebrated approximation algorithms with provable performance guarantees proceed by iterative update methods coupled with bounds on the local change of the objective value (e.g., basis reduction in Lenstra, Lenstra, & Lovasz, 1981, and typical primal-dual methods in Vazirani, 2002).

Encouraged both by the possibility of utilizing such heuristics in practice and the natural question of understanding the structural properties of heuristics (and search spaces) that indeed guarantee palatable performance on the part of A^*, we study the behavior of A^* when provided with a heuristic function that is an ϵ-approximation to the cost of a cheapest path to a solution. As certain natural situations arise where approximation quality is asymmetric (i.e., the case of an admissible heuristic), we slightly refine the notion of accuracy by distinguishing the multiplicative factors in the two sides of an approximation: we say that a heuristic h is an *(ϵ_1, ϵ_2)-approximation* to the actual cost function h^*, or simply *(ϵ_1, ϵ_2)-approximate*, if $(1-\epsilon_1)h^* \leq h \leq (1+\epsilon_2)h^*$. In particular, admissible heuristics with ϵ-approximation are $(\epsilon, 0)$-approximate. We will call a heuristic *δ-accurate* if it is (ϵ_1, ϵ_2)-approximate and $\delta = \epsilon_1 + \epsilon_2$. A detailed description appears in Section 2.1.

1.1 A Sketch of the Results

We initially model our search space as an infinite b-ary tree with a distinguished root. A problem instance is determined by a set S of nodes of the tree—the "solutions" to the problem. The cost associated with a solution $s \in S$ is simply its depth. The search procedure is equipped with *(i.)* an oracle which, given a node n, determines if $n \in S$, and *(ii.)* an *heuristic* function h, which assigns to each node n of the tree an estimate of the actual length $h^*(n)$ of the shortest (descending) path to a solution. Let S be a solution set in which the first (and hence optimal) solution appears at depth d. We establish a family of upper bounds on the number of nodes expanded by A^*: if h is an (ϵ_1, ϵ_2)-approximation of h^*, then A^* finds a solution of cost no worse than $(1+\epsilon_2)d$ and expands no more than $2b^{(\epsilon_1+\epsilon_2)d} + dN_{\epsilon_1+\epsilon_2}$ nodes, where N_δ denotes the number of solutions at depth less than $(1+\delta)d$. See Lemma 3.1 below for stronger results. We emphasize that this bound applies to any solution space and can be generalized to search models with non-uniform branching factors and non-uniform edge costs (see Section 5).

We go on to show that this upper bound is essentially tight; in fact, we show that the bound is nearly achieved even by *non-adversarially determined* solution spaces selected according to a simple stochastic rule (see Theorems 3.1 and 4.1.). We remark that these bounds on running time fall off rapidly as the accuracy of the heuristics increases, as long as the number of near-optimal solutions is not too large (although it may grow exponentially). For instance, the effective branching factor of A^* guided by an admissible δ-accurate heuristic will be reduced to b^δ if $N_\delta = O(b^{\delta d})$. However, in the worst cases, which occur when the search space has an overwhelming number of near-optimal solutions, A^* still has to expand almost as many nodes as brute-force does, regardless of heuristic accuracy. Likewise, strong guarantees on $\delta < 1$ are, in general, necessary to effect appreciable changes in average branching factor. This is discussed in Theorem 4.2.

After establishing bounds for the tree-based search model, we examine the time complexity of A^* on a graph by "unrolling" the graph into an equivalent tree and then bounding the number of near-optimal solutions in the tree which are a "lift" of a solution in the original graph. This appears in Section 6. Using spectral graph theory, we show that the number N_δ of lifted solutions on the tree corresponding to a b-regular graph G is $O(\mu^{(1+\delta)d})$, assuming the optimal solution depth d is $O(\log_b |G|)$ and the number solutions in G is constant, where μ is the *second* largest eigenvalue (in absolute value) of the adjacency matrix of G. In particular, for almost all b-regular graphs in which b does not grow with the size of graphs, we have $\mu \leq 2\sqrt{b}$, which yields the *effective branching factor* of A^* search on such graphs is roughly at most $8b^{(1+\delta)/2}$ if the heuristic is δ-accurate. We also experimentally evaluate these heuristics.

Experimental Results and the Relationship to A^* in Practice. Of course, these upper bounds are most interesting if they reflect the behavior of search problems in practice. The bounds above guarantee, in general, that E, the number of nodes expanded by A^* with a δ-accurate heuristic, satisfies

$$E \leq 2b^{\delta d} + dN_\delta \,.$$

Under the plausible condition that $N_\delta \approx b^{\delta d}$, we have simply $E \approx cb^{\delta d}$ node expansions for a constant c that does not depend on δ (c may depend on k and/or other properties of the search space). This suggests the hypothesis that for hard combinatorial problems with suitably sparse near-optimal solutions,

$$E \approx cb^{\delta d} \quad \text{or, equivalently,} \quad \log E \approx \log c + \delta d \log b \,. \tag{1}$$

In particular, this suggests a linear dependence of $\log E$ on δ.

To explore this hypothesis, we conducted a battery of experiments on the natural search-tree presentation of the well-studied *Knapsack Problem*. Here we obtain an admissible δ-accurate heuristic by applying the Fully Polynomial Time Approximation Scheme (FPTAS) for the problem due to the work of Ibarra and Kim (1975) (see also Vazirani, 2001, p. 70), which provides us with a convenient method for varying δ without changing the other parameters of the search. We remark that the natural search space for the problem is a quite irregular edge-weighted directed graph on which A^* can avoid reopening any node. Thus, this search space is equivalent to one of its spanning subtrees in terms of A^*'s behaviors. In order to focus on computationally nontrivial examples, we generate Knapsack instances from distributions that are empirically hard for the best known exact algorithms (Pisinger, 2005). The results of these experiments yield remarkably linear behavior (of $\log E$ as a function of δ) for a quite wide window of values: indeed, our tests yield R^2 correlation coefficients (of the least-square linear regression model) in excess of 90% with δ in the range $(.5, 1)$ for most Knapsack instances. See Section 7.1 for details.

While the experimental results discussed above for the Knapsack problem support the linear scaling of (1), several actual parameters of the search are unknown: for example, we cannot rule out the possibility that the approximation algorithm, when asked to produce an ϵ-approximation, does not in fact produce a *significantly better* approximation. While this seems far-fetched, such behavior could provide spurious evidence for linear scaling. To explore the hypothesis in more detail, we additionally explore a more artificial search space for the *partial Latin square completion (PLS) problem* in which we can provide precise control of δ (and, in fact, N_δ). The PLS problem is featured in a number of benchmarks for local search and complete search methods. Roughly, this is the problem of finding an assignment of values to the empty cells of a partially filled $n \times n$ table so that each row and column in the completed table is a permutation of the set $\{1, \ldots, n\}$. In our formulation of the problem, the search space is a $2n$-regular graph, thus the brute-force branching factor is $2n$. On this search space, by controlling N_δ, we prove an asymptotic upper bound of $(1+\delta)\left(1+{}^{1}\!/\!{}_{\delta}\right)^{\delta} n^{\delta}$ on the effective branching factor of A^* coupled with *any* δ-accurate heuristic. We also experimentally evaluate the effective branching factor of A^* with the admissible δ-accurate heuristic $(1-\delta)h^*$, with which A^* expands more nodes than with any admissible δ-accurate heuristic strictly larger than $(1-\delta)h^*$.

We remark that while the PLS problem itself is well-studied and natural, we invent specific search space structure on the problem that allows us to analytically control the number of near-optimal solutions. Unlike the Knapsack problem, where we can construct an efficient admissible δ-accurate heuristic for every fixed δ thanks to the given FPTAS, known approximation algorithms for the PLS problem are much weaker—they provide approximations for specific constants $(1/e)$. To avoid this hurdle, we construct instances of PLS with *known* solution, from which we extract the heuristics $(1-\delta)h^*$. Despite these "planted" solutions and contrived heuristics, the infrastructure provides an example of a combinatorially rich search space with known solution multiplicity and a heuristic of *known quality*, and so provides a means for experimentally measuring the relationship between heuristic accuracy and running time. Our empirical data results in remarkable agreement with the theoretical upper bounds. More subtly, by empirically analyzing the linear dependence of $\log E$ on δ, we see that the effective branching factor of A^* using the heuristic $(1-\delta)h^*$ on the given PLS search space is roughly $(2n)^{0.8\delta}$; see Section 7.2.

As far as we are aware, these are the first experimental results that explore the relationship between δ and E. Understanding heuristic accuracy and solution space structure in general (and the ensuing bounds on A^* running time) for problems and heuristics of practical interest remains an intriguing open problem. We remark that for problems such as the (n^2-1)-puzzle, which have been extensively used as test cases for A^*, it seems difficult to find heuristics with accuracy sufficient to significantly reduce average branching factor. The best rigorous algorithms can only give rather large constant guarantees (Ratner & Warmuth, 1990; Parberry, 1995): in particular, Parberry (1995) shows that one can quickly compute solutions (and hence approximate heuristics) that are no more than a factor 19 worse than optimal; the situation is somewhat better for random instances, where he establishes a 7.5-factor. See Demaine's (2001) work for a general discussion.

Observe that any search algorithm *not privy to heuristic information* requires $\Omega(b^d)$ running time, in general, to find a solution. High probability statements of the same kind can be made if the solution space is selected from a sufficiently rich family. Such pessimistic lower bounds exist even in situations where the search space is highly structured (Aaronson, 2004). Our results suggest that accurate heuristic information can have a dramatic impact on A^* search, even in face of substantial solution multiplicity.

This article expands the conference article (Dinh, Russell, & Su, 2007) where the complexity of A^* with an ϵ-approximate heuristic function was studied over trees. In this article, we generalize this to asymmetric approximation, develop analogous bounds over general search spaces, establishing a connection to algebraic graph theory, and report on a battery of supporting experimental results.

1.2 Motivation and Related Work

The A^* algorithm has been the subject of an enormous body of literature, often investigating its behavior in relation to a specific heuristic and search problem combination, (e.g., Zahavi, Felner, Schaeffer, & Sturtevant, 2007; Sen, Bagchi, & Zhang, 2004; Korf & Reid, 1998; Korf, Reid, & Edelkamp, 2001; Helmert & Röger, 2008). Both space complexity (Korf, 1985) and time complexity have been addressed at various levels of abstraction. Abstract formulations, involving accuracy guarantees like those we consider, have been studied, but only in tree models where the search space possesses a *single* solution. In this single solution framework, Gaschnig (1979) has given exponential lower bounds of $\Omega(b_\delta^d)$ on the time complexity for admissible δ-accurate heuristics, where $b_\delta \stackrel{\text{def}}{=} b^{\delta/(2-\delta)} \leq b^\delta$ (see also Pearl, 1984, p. 180), while Pohl (1977) has studied more restrictive (additive) approximation guarantees on h which result in linear time complexity. Average-case analysis of A^* based on probabilistic accuracy of heuristics has also been given for single-solution search spaces (Huyn, Dechter, & Pearl, 1980). These previous analysis suggested that the effect of heuristic functions would reduce the effective branching factor of the search, which is consistent with our results when applied to the single-solution model (the special case when $N_\delta = 1$ for all $\delta > 0$). The single solution model, however, appears to be an inappropriate abstraction of most search problems featuring multiple solutions, as it has been recognized that "*. . . the presence of multiple solutions may significantly deteriorate A^*'s ability to benefit from improved precision.*" (Pearl, 1984, p. 192) (emphasis added).

The problem of understanding the time complexity in terms of structural properties of h on multiple-solution spaces has been studied by Korf and Reid (1998), Korf et al. (2001), and Korf (2000), using an estimate based on the distribution of $h(\cdot)$ values. In particular, they studied an abstract search space given by a b-ary tree and concluded that "the effect of a heuristic function is to reduce the effective depth of a search rather than the effective branching factor" (Korf & Reid, 1998; Korf et al., 2001). For the case of *accurate* heuristics with controlled solution multiplicity, this conclusion directly contradicts our findings, which indicate dramatic reduction in effective branching factor for such cases. To explain this discrepancy, we observe that their analysis relies on an "equilibrium assumption" that fails for accurate heuristics (in fact, it fails even for much weaker heuristic guarantees, such as $h(v) \geq \epsilon h^*(v)$ for small $\epsilon > 0$). The basic structure of their argument, however, can be naturally adapted to the case of accurate heuristics, in which case it yields a reduction in effective branching factor. We give a detailed discussion in Section 8.

As a follow-up to Korf and Reid (1998), Korf et al. (2001), and Korf's (2000) work, Edelkamp (2001) examined A^* (indeed, IDA*) on undirected graphs, relying on the equilibrium assumption. Edelkamp's new technique is the use of graph spectrum to estimate the number $n^{(\ell)}$ of nodes at certain depth ℓ in the brute-force search tree (same as our cover tree). However, unlike our spectral analysis, which is of the original search graph G, Edelkamp analyzed the spectrum of a related "equivalence graph," which has quite different structural properties. Specifically, Edelkamp found that the asymptotic branching factor, defined by the ratio $n^{(\ell)}/n^{(\ell-1)}$ for large ℓ, equals the *largest* eigenvalue of the adjacency matrix of the equivalence graph for certain Puzzle problems. To compare, our spectral analysis depends on the *second* largest eigenvalue of the adjacency matrix A_G of the original search graph G, while the largest eigenvalue of A_G always equals the branching factor, assuming G is regular.

Additionally, the analyses of Korf and Reid (1998), Korf et al. (2001), and Korf (2000) (and therefore, of Edelkamp, 2001) focus on a particular subclass of admissible heuristics, called *consistent heuristics*. We remark that the heuristics used in our experiments for the Knapsack problem are admissible but likely inconsistent. Zhang, Sturtevant, Holte, Schaeffer, and Felner (2009) and Zahavi et al. (2007) discuss usages of inconsistent heuristics in practice.

Our work below explores both worst-case and average-case time complexity of A^* search on both trees and graphs with multiple solutions when coupled with heuristics possessing accuracy guarantees. We make no assumptions regarding consistency or admissibility of the heuristics, though several of our results can be naturally specialized to this case. In addition to studying the effect of heuristic accuracy, our results also shed light on the sensitivity of A^* to the distribution of solutions and the combinatorial structure of the underlying search spaces (e.g., graph eigenvalues, which measure, among other things, the extent of connectedness for graphs). As far as we are aware, these are the first rigorous results combining search space structure and heuristic accuracy in a single framework for predicting the behavior of A^*.

2. Preliminaries

A typical search problem is defined by a search graph with a starting node and a set of goal nodes called solutions. Any instance of A^* search on a graph, however, can be simulated by A^* search on a cover tree without reducing running time; this is discussed in Section 6.1. Since the number of expansions on the cover tree of a graph is larger than or equal to that on the original graph, it is sufficient to upper bound the running time of A^* search on the cover tree. With this justification, we begin with considering the A^* algorithm for search problems on a rooted tree.

Problem Definition and Notations. Let $\mathcal{T}$ be a tree representing an infinite search space, and let r denote the root of $\mathcal{T}$. For convenience, we also use the symbol $\mathcal{T}$ to denote the set of vertices in the tree $\mathcal{T}$. Solutions are specified by a nonempty subset $S \subset \mathcal{T}$ of nodes in $\mathcal{T}$. Each edge on $\mathcal{T}$ is assigned a positive number called the *edge cost*. For each vertex v in $\mathcal{T}$, let

- $\textsc{SubTree}(v)$ denote the subtree of $\mathcal{T}$ rooted at v,
- $\textsc{Path}(v)$ denote the path in $\mathcal{T}$ from root r to v,
- $g(v)$ denote the total (edge) cost of $\textsc{Path}(v)$, and
- $h^*(v)$ denote the cost of the least costly path from v to a solution in $\textsc{SubTree}(v)$. (We write $h^*(v) = \infty$ if no such solution exists.)

The objective value of this search problem is $h^*(r)$, the cost of the cheapest path from the root r to a solution. The cost of a solution $s \in S$ is the value of $g(s)$. A solution of cost equal to $h^*(r)$ is referred to as *optimal*.

The A^* algorithm is a best-first search employing an additive evaluation function $f(v) = g(v) + h(v)$, where h is a function on $\mathcal{T}$ that heuristically estimates the actual cost h^*. Given a heuristic function $h : \mathcal{T} \to [0, \infty]$, the A^* algorithm using h for our defined search problem on the tree $\mathcal{T}$ is described as follows:

Algorithm 1 A^* search on a tree

1. Initialize $\textsc{Open} := \{r\}$.
2. Repeat until $\textsc{Open}$ is empty:
   (a) Remove from $\textsc{Open}$ a node v at which the function $f = g + h$ is minimum.
   (b) If v is a solution, exit with success and return v.
   (c) Otherwise, expand node v, adding all its children in $\mathcal{T}$ to $\textsc{Open}$.
3. Exit with failure.

It is known (e.g., Dechter & Pearl, 1985, Lemma 2) that at any time before A^* terminates, there is always a vertex v present in $\textsc{Open}$ such that v lies on a solution path and $f(v) \leq M$, where M is the min-max value defined as follows:

$$M \stackrel{\text{def}}{=} \min_{s \in S} \left(\max_{u \in \textsc{Path}(s)} f(u) \right). \tag{2}$$

This fact leads to the following node expansion conditions:

- Any vertex v expanded by A^* (with heuristic h) must have $f(v) \leq M$. (cf., Dechter & Pearl, 1985, Thm. 3). We say that a vertex v satisfying $f(v) \leq M$ is *potentially expanded* by A^*.
- Any vertex v with
$$\max_{u \in \textsc{Path}(v)} f(u) < M$$
must be expanded by A^* (with heuristic h) (cf., Dechter & Pearl, 1985, Thm. 5). In particular, when the function f monotonically increases along the path from the root r to v, the node v must be expanded if $f(v) < M$.

The value of M will be obtained on the solution path with which A^* search terminates (Dechter & Pearl, 1985, Lemma 3), which implies that M is an upper bound for the cost of the solution found by the A^* search.

We remark that if h is a reasonable approximation to h^* along the path to the optimal solution, this immediately provides some control on M. In particular:

Proposition 2.1. *(See also Davis, Bramanti-Gregor, & Wang, 1988) Suppose that for some $\alpha \geq 1$, $h(v) \leq \alpha h^*(v)$ for all vertices v lying on an optimal solution path; then $M \leq \alpha h^*(r)$.*

Proof. Let s be an optimal solution. For all $v \in \textsc{Path}(s)$,

$$f(v) \leq g(v) + \alpha h^*(v) = g(v) + \alpha(g(s) - g(v)) \leq \alpha g(s)\,.$$

Hence $M \leq \max_{v \in \textsc{Path}(s)} f(v) \leq \alpha g(s) = \alpha h^*(r)$. □

In particular, $M = h^*(r)$ if the heuristic function satisfies $h(v) \leq h^*(v)$ for all $v \in \mathcal{T}$, in which case the heuristic function is called *admissible*. The observation above recovers the fact that A^* always finds an optimal solution when coupled with an admissible heuristic function (cf., Pearl, 1984, Thm. 2, §3.1). Admissible heuristics also possess a natural *dominance* property (Pearl, 1984, Thm. 7, p. 81): for any admissible heuristic functions h_1 and h_2 on $\mathcal{T}$, if h_1 is *more informed than* h_2, i.e., $h_1(v) > h_2(v)$ for all $v \in \mathcal{T} \backslash S$, then A^* using h_1 *dominates* A^* using h_2, i.e., every node expanded by A^* using h_1 is also expanded by A^* using h_2.

2.1 Approximate Heuristics

Recall from the introduction that we shall focus on heuristics providing an (ϵ_1, ϵ_2)-approximation to the actual optimal cost to reach a solution:

Definition. Let $\epsilon_1, \epsilon_2 \in [0, 1]$. A heuristic function h is called (ϵ_1, ϵ_2)-*approximate* if

$$(1 - \epsilon_1)h^*(v) \leq h(v) \leq (1 + \epsilon_2)h^*(v) \quad \text{for all } v \in \mathcal{T}.$$

An (ϵ_1, ϵ_2)-approximate heuristic is simply called ϵ-*approximate* if both $\epsilon_1 \leq \epsilon$ and $\epsilon_2 \leq \epsilon$. If a heuristic function h is (ϵ_1, ϵ_2)-approximate, we shall say that h has a *heuristic error* $\epsilon_1 + \epsilon_2$, or h is $(\epsilon_1 + \epsilon_2)$-*accurate*.

As we will see below, these two approximation factors control the performance of A^* search in rather different ways: while ϵ_1 only effects the running time of A^*, ϵ_2 has impact on both the running time and the quality of the solution found by A^*. Particularly, the special case $\epsilon_2 = 0$ corresponds to admissible heuristics, with which A^* always finds an optimal solution. In general, by Proposition 2.1, we have:

Fact 1. *If h is (ϵ_1, ϵ_2)-approximate, then $M \leq (1 + \epsilon_2)h^*(r)$.*

Hence, the solution found by A^* using an (ϵ_1, ϵ_2)-approximate heuristic must have cost no more than $(1 + \epsilon_2)h^*(r)$ and thus exceeds the optimal cost by no more than a multiplicative factor equal ϵ_2.

Definition. Let $\delta \geq 0$. A solution of cost less than $(1 + \delta)h^*(r)$ is called a δ-*optimal solution.*

Assumptions. To simplify the analysis for now, we assume that the search tree $\mathcal{T}$ is b-ary and that every edge is of unit cost unless otherwise specified. In this case, the cost $g(v)$ is simply the depth of node v in $\mathcal{T}$ and $h^*(v)$ is the shortest distance from v to a solution that is a descendant of v. Throughout, the parameters $b \geq 2$ (the branching factor of the search space) and $\epsilon_1 \in (0, 1], \epsilon_2 \in [0, 1]$ (the quality of the approximation provided by the heuristic function) are fixed. We rule out the case $\epsilon_1 = 0$ for simplicity.

3. Upper Bounds on Running Time of A^* on Trees

We are now going to establish upper bounds on the running time of A^* search on the tree model. We will first show a generic upper bound that applies to any solution space. We then apply this generic upper bound to a natural stochastic solution space model.

3.1 A Generic Upper Bound

As mentioned in the introduction, we begin with an upper bound on the time complexity of A^* search depending only on the "weight distribution" of the solution set, in addition to the heuristic's approximation factors. We shall, in fact, upper bound the number of potentially expanded nodes, which is clearly an upper bound on the number of nodes actually expanded by A^*:

Lemma 3.1. *Let S be a solution set whose optimal solutions lie at depth d. Then, for every $\gamma \geq 0$, the number of nodes expanded by A^* search on the tree $\mathcal{T}$ with an (ϵ_1, ϵ_2)-approximate heuristic is no more than*

$$2b^{(\gamma\epsilon_1+\epsilon_2+1-\gamma)d} + \gamma(1-\epsilon_1)dN_{\gamma\epsilon_1+\epsilon_2}$$

nodes, where N_δ is the number of δ-optimal solutions.

The presence of the independent parameter γ offers a flexible way to apply the upper bound in Lemma 3.1. In particular, applying Lemma 3.1 with $\gamma = 1$ and using the fact that $1 - \epsilon_1 \leq 1$, we arrive at the upper bound of $2b^{(\epsilon_1+\epsilon_2)d} + dN_{\epsilon_1+\epsilon_2}$ mentioned in the introduction. This bound works best when[1] $N_{\epsilon_1+\epsilon_2} = \Theta(b^{(\epsilon_1+\epsilon_2)d})$. In general, if $N_{\epsilon_1+\epsilon_2} = O(b^{(\epsilon_1+\epsilon_2)d})$, we should choose the least $\gamma \geq 1$ for which $N_{\gamma\epsilon_1+\epsilon_2} = O(b^{(\epsilon_1+\epsilon_2)d})$. In the opposite case, if $N_{\epsilon_1+\epsilon_2} = \Omega(b^{(\epsilon_1+\epsilon_2+c)d})$ for some positive constant $c \leq 1 - \epsilon_1$, we can obtain a better bound by choosing $\gamma = 1 - c/(1-\epsilon_1) < 1$, since $N_{\epsilon_1+\epsilon_2}$ dominates both terms $\Omega(b^{(\gamma\epsilon_1+\epsilon_2+1-\gamma)d})$ and $N_{\gamma\epsilon_1+\epsilon_2}$ given such a choice of γ.

Proof of Lemma 3.1. Let $d = h^*(r)$ and let $\delta = \gamma\epsilon_1 + \epsilon_2$. Consider a node v which does not lie on any path from the root to a δ-optimal solution, so that $h^*(v) \geq (1+\delta)d - g(v)$. Then

$$f(v) \geq g(v) + (1-\epsilon_1)[(1+\delta)d - g(v)] = (1-\epsilon_1)(1+\delta)d + \epsilon_1 g(v)\,.$$

Recall that a node is potentially expanded by A^* if its f-value is less than or equal to M. Since $M \leq (1+\epsilon_2)d$, the node v will not be potentially expanded if

$$(1-\epsilon_1)(1+\delta)d + \epsilon_1 g(v) > (1+\epsilon_2)d\,. \tag{3}$$

Since $\epsilon_1 > 0$, the inequality (3) is equivalent to

$$g(v) > (\epsilon_2/\epsilon_1 - \delta/\epsilon_1 + 1 + \delta)d = (\gamma\epsilon_1 + \epsilon_2 + 1 - \gamma)d\,.$$

In other words, any node at depths in the range

$$\big((\gamma\epsilon_1 + \epsilon_2 + 1 - \gamma)d,\ (1+\epsilon_2)d\big]$$

can be potentially expanded only when it lies on the path from the root to some δ-optimal solution. On the other hand, on each δ-optimal solution path, there are at most $\gamma(1-\epsilon_1)d$ nodes at depths in $\big((\gamma\epsilon_1+\epsilon_2+1-\gamma)d, (1+\epsilon_2)d\big]$. Pessimistically assuming that *all* nodes with depth no more than $(\gamma\epsilon_1+\epsilon_2+1-\gamma)d$ are potentially expanded in addition to those on paths to δ-optimal solutions yields the statement of the lemma. (Note that as $b \geq 2$, $\sum_{i=0}^{\ell} b^i \leq 2b^\ell$ and that every potentially expanded node v must have depth $g(v) \leq f(v) \leq M \leq (1+\epsilon_2)d$.) $\square$

1. Recall some asymptotic notations: $f(n) = \Theta(g(n))$ means there exist constants $c_1, c_2 > 0$ such that $c_1 g(n) \leq f(n) \leq c_2 g(n)$ for sufficiently large n; $f(n) = \Omega(g(n))$ means there exists a constant $c > 0$ such that $cg(n) \leq f(n)$ for sufficiently large n.

3.2 An Upper Bound on a Natural Search Space Model

While actual time complexity will depend, of course, on the precise structure of S and h, we show below that this bound is essentially tight for a rich family of solution spaces. We consider a sequence of search problems of "increasing difficulty," expressed in terms of the depth d of the optimal solution.

A Stochastic Solution Space Model. For a parameter $p \in [0,1]$, consider the solution set S which is obtained by independently placing each node of $\mathcal{T}$ into S with probability p. In this setting, S is a random variable and is written S_p. When solutions are distributed according to S_p, observe that the expected number of solutions at depth d is precisely pb^d and that when $p = b^{-d}$ an optimal solution lies at depth d with constant probability. For this reason, we focus on the specific values $p_d = b^{-d}$ and consider the solution set S_{p_d} for each $d > 0$. Recall that under this model, it is likely for the optimal solutions to lie at depth d and, more generally, we can see that with very high probability the optimal solutions of any particular subtree will be located near depth d (with respect to the root of the subtree). We make this precise below.

Lemma 3.2. *Suppose the solutions are distributed according to S_{p_k}. Then for any node $v \in \mathcal{T}$ and $t > 0$,*

$$1 - 2b^{t-d} \le \Pr[h^*(v) > t] \le e^{-b^{t-d}} .$$

Proof. In the tree $\textsc{SubTree}(v)$, there are $n = \sum_{i=0}^{t} b^i = (b^{t+1}-1)/(b-1)$ nodes at depths t or less, so $\Pr[h^*(v) > t] = (1 - b^{-d})^n$. We have

$$1 - nb^{-d} \le (1 - b^{-d})^n \le \exp\left(-nb^{-d}\right) .$$

The first inequality is obtained by applying Bernoulli's inequality, and the last one is implied from the fact that $1 - x \le e^{-x}$ for all x. Observing that

$$b^t \le \frac{b^{t+1}-1}{b-1} \le 2b^t$$

for $b \ge 2$ completes the proof. □

Observe that in the S_{p_d} model, conditioned on the likely event that the optimal solutions appear at depth d, the expected number of δ-optimal solutions is $\Theta(b^{\delta d})$. In this situation, according to Lemma 3.1, A^* expands no more than $O(b^{(\gamma\epsilon_1+\epsilon_2+1-\gamma)d}) + O(db^{(\gamma\epsilon_1+\epsilon_2)d})$ vertices in expectation, for any $\gamma \ge 0$. The leading exponential term in this bound is equal to

$$\max\left\{(\gamma\epsilon_1 + \epsilon_2 + 1 - \gamma)d,\ (\gamma\epsilon_1 + \epsilon_2)d\right\} ,$$

which is minimal when $\gamma = 1$. This suggests the best upper bound that can be inferred from the family of bounds in Lemma 3.1 is $\operatorname{poly}(d)b^{(\epsilon_1+\epsilon_2)d}$ (for S_{p_d}).

Before discussing the average-case time complexity of A^* search, we record the following well-known Chernoff bound, which will be used to control the tail bounds in our analysis later.

Lemma 3.3 (Chernoff bound, Chernoff, 1952)**.** *Let Z be the sum of mutually independent indicator random variables with expected value $\mu = \mathbb{E}[Z]$. Then for any $\lambda > 0$,*

$$\Pr[Z > (1+\lambda)\mu] < \left[\frac{e^\lambda}{(1+\lambda)^{1+\lambda}}\right]^\mu .$$

A detailed proof can be found in the book of Motwani and Raghavan (1995). In several cases below, while we do not know exactly the expected value of the variable to which we wish to apply the tail bound in Lemma 3.3, we can compute sufficiently good upper bounds on the expected value. In order to apply the Chernoff bound in such a case, we actually require a monotonicity argument: If $Z = \sum_{i=1}^n X_i$ and $Z' = \sum_{i=1}^n X'_i$ are sums of independent and identically distributed (i.i.d.) indicator random variables so that $\mathbb{E}[X_i] \le \mathbb{E}[X'_i]$, then $\Pr[Z > \alpha] \le \Pr[Z' > \alpha]$ for all α. With this argument and by applying Lemma 3.3 for $\lambda = e - 1$, we obtain:

Corollary 3.1. *Let Z be the sum of n i.i.d. indicator random variables so that $\mathbb{E}[Z] \le \mu \le n$, then*

$$\Pr[Z > e\mu] < e^{-\mu}.$$

Adopting the search space whose solutions are distributed according to S_{p_d}, we are ready to bound the running time of A^* on average when guided by an (ϵ_1, ϵ_2)-approximate heuristic:

Theorem 3.1. *Let d be sufficiently large. With probability at least $1 - e^{-d} - e^{-2d^3}$, A^* search on the tree $\mathcal{T}$ using an (ϵ_1, ϵ_2)-approximate heuristic function expands no more than $12d^4 b^{(\epsilon_1+\epsilon_2)d}$ vertices when solutions are distributed according to the random variable S_{p_d}.*

Proof. Let X be the random variable equal to the total number of nodes expanded by the A^* with an (ϵ_1, ϵ_2)-approximate heuristic. Of course the exact value of, say, $\mathbb{E}[X]$ depends on h; we will prove upper bounds achieved with high probability for any (ϵ_1, ϵ_2)-approximate h. Applying Lemma 3.1 with $\gamma = 1$, we conclude

$$X \le 2b^{(\epsilon_1+\epsilon_2)h^*(r)} + (1-\epsilon_1)h^*(r)N_{\epsilon_1+\epsilon_2}.$$

Thus it suffices to control both $h^*(r)$ and the number $N_{\epsilon_1+\epsilon_2}$ of $(\epsilon_1 + \epsilon_2)$-optimal solutions.

We will utilize the fact that in the S_{p_d} model, the optimal solutions are unlikely to be located far from depth d. To this end, let E_{far} be the event that $h^*(r) > d + \Delta$ for some $\Delta < d$ to be set later. Lemma 3.2 immediately gives $\Pr[E_{\text{far}}] \le e^{-b^{\Delta}}$.

Observe that conditioned on $\overline{E_{\text{far}}}$, we have $h^*(r) \le d+\Delta$ and $N_{\epsilon_1+\epsilon_2} \le Z$, where Z is the random variable equal to the number of solutions with depth no more than $(1 + \epsilon_1 + \epsilon_2)(d + \Delta)$. We have

$$\mathbb{E}[Z] \le b^{-d} \cdot 2b^{(1+\epsilon_1+\epsilon_2)(d+\Delta)} = 2b^{(\epsilon_1+\epsilon_2)d+(1+\epsilon_1+\epsilon_2)\Delta} < 2b^{(\epsilon_1+\epsilon_2)d+3\Delta}$$

and, applying the Chernoff bound in Corollary 3.1 to control Z,

$$\Pr\Big[Z > 2eb^{(\epsilon_1+\epsilon_2)d+3\Delta}\Big] \le \exp\left(-2b^{(\epsilon_1+\epsilon_2)d+3\Delta}\right) \le e^{-2b^{3\Delta}}.$$

Letting E_{thick} be the event that $Z \ge 6b^{(\epsilon_1+\epsilon_2)d+3\Delta}$, observe

$$\Pr[E_{\text{thick}}] \le \Pr\Big[Z > 2eb^{(\epsilon_1+\epsilon_2)d+3\Delta}\Big] \le e^{-2b^{3\Delta}}.$$

To summarize: when neither E_{far} nor E_{thick} occurs,

$$\begin{aligned} X &\le 2b^{(\epsilon_1+\epsilon_2)(d+\Delta)} + (1-\epsilon_1)(d+\Delta)6b^{(\epsilon_1+\epsilon_2)d+3\Delta} \\ &\le 6(d+\Delta)b^{(\epsilon_1+\epsilon_2)d+3\Delta} \\ &\le 12db^{(\epsilon_1+\epsilon_2)d+3\Delta}. \end{aligned}$$

Hence,

$$\Pr\Big[X > 12db^{(\epsilon_1+\epsilon_2)d+3\Delta}\Big] \le \Pr[E_{\text{far}} \vee E_{\text{thick}}] \le e^{-b^{\Delta}} + e^{-2b^{3\Delta}}.$$

To infer the bound stated in our theorem, set $b^{\Delta} = d$ so that $b^{(\epsilon_1+\epsilon_2)d+3\Delta} = d^3 b^{(\epsilon_1+\epsilon_2)d}$, completing the proof. □

Remark By similar methods, other trade-offs between the error probability and the resulting bound on the number of expanded nodes can be obtained.

4. Lower Bounds on Running Time of A^* on Trees

We establish that the upper bounds in Theorem 3.1 are tight to within a $O(1/\sqrt{d})$ term in the exponent. We begin by recording the following easy fact about solution distances in this discrete model.

Fact 2. *Let $\Delta \leq h^*(r)$ be a nonnegative integer. Then for every solution s, there is a node $v \in$ $\textsc{Path}(s)$ such that $h^*(v) = \Delta$.*

Proof. Fix a distance $\Delta \leq h^*(r)$. We will prove the lemma by induction on the depth of solutions. The lemma clearly holds for optimal solutions. Consider a solution s which may not be optimal, and let $v \in \textsc{Path}(s)$ be the node which is Δ level far from s so that $h^*(v) \leq \Delta$. If $h^*(v) < \Delta$, there must be another solution $s' \in \textsc{SubTree}(v)$ that is closer to v. By the induction assumption, there is a node $v' \in \textsc{Path}(s')$ with $h^*(v') = \Delta$. This node v' must be an ancestor of v, since the distance between v and s' is less than Δ while the distance between v' and s' is at least Δ, completing the proof. □

We proceed now to the lower bound.

Theorem 4.1. *Let d be sufficiently large. For solutions distributed according to S_{p_d}, with probability at least $1 - b^{-\sqrt{d}}$, there exists an (ϵ_1, ϵ_2)-approximate heuristic function h so that the number of vertices expanded by A^* search on the tree $\mathcal{T}$ using h is at least $b^{(\epsilon_1+\epsilon_2)d-4\sqrt{d}}/8$.*

Proof. Our plan is to define a pathological heuristic function that forces A^* to expand as many nodes as possible. Note that the heuristic function here is allowed to overestimate h^*. Intuitively, we wish to construct a heuristic function that overestimates h^* at nodes close to a solution and underestimates h^* at nodes far from solutions, leading A^* astray whenever possible. Recall that for every vertex v, it is likely to have a solution lying at depth d of $\textsc{SubTree}(v)$. Thus we can use the quantity $h^*(v) \leq d - \Delta$ to formalize the intuitive notion that the node v is close to a solution, where the quantity $\Delta < d$ will be determined later. Our heuristic function h is formally defined as follows:

$$h(v) = \begin{cases} (1+\epsilon_2)h^*(v) & \text{if } h^*(v) \leq d - \Delta, \\ (1-\epsilon_1)h^*(v) & \text{otherwise.} \end{cases}$$

Observe that the chance for a node to be overestimated is small since, by Lemma 3.2,

$$\Pr[v \text{ is overestimated}] = \Pr[h^*(v) \leq d - \Delta] \leq 2b^{-\Delta} \tag{4}$$

for any node v. Also note that if a node v does not have any overestimated ancestor, then the f values will monotonically increase along the path from root to v.

Naturally, we also wish to ensure that the optimal solution is not too close to the root. Let E_{close} be the event that $h^*(r) \leq d - \Delta$. Again by Lemma 3.2,

$$\Pr[E_{\text{close}}] \leq 2b^{-\Delta}\,.$$

We then will see that conditioned on the event $\overline{E_{\text{close}}}$, which means "$h^*(r) > d - \Delta$," every solution will be "obscured" by an overestimated node that is *not too close* to a solution. Concretely, up to issues of integrality, Fact 2 asserts that for every solution s, there must be a node v on the path from the root to s with $h^*(v) = d - \Delta$, as long as $d - \Delta < h^*(r)$.

Assume $\overline{E_{\text{close}}}$: then whenever $h^*(v) = d - \Delta$, we have $g(v) \geq h^*(r) - (d - \Delta) > 0$ and $h(v) = (1+\epsilon_2)(d - \Delta)$, and thus $f(v) > (1+\epsilon_2)(d-\Delta)$. Since every solution is "obscured" by some overestimated node whose f value is larger than $(1+\epsilon_2)(d-\Delta)$, we have $M > (1+\epsilon_2)(d-\Delta)$, where M is the min-max value defined in (2). It follows that a node v must be expanded if $\textsc{Path}(v)$ does not contain any overestimated node and $f(v) \le (1+\epsilon_2)(d-\Delta)$. When $\textsc{Path}(v)$ does not contain an overestimated node, we have $f(v) = g(v) + (1-\epsilon_1)h^*(v)$, so

$$f(v) \le (1+\epsilon_2)(d-\Delta) \Leftrightarrow (1-\epsilon_1)h^*(v) \le (1+\epsilon_2)(d-\Delta) - g(v)\,,$$

since $\epsilon_1 < 1$. Therefore, we say a node v is *required* if there is no overestimated node in $\textsc{Path}(v)$ and $(1-\epsilon_1)h^*(v) \le (1+\epsilon_2)(d-\Delta) - g(v)$. To recap, conditioned on $\overline{E_{\text{close}}}$, the set of required nodes is a subset of the set of nodes expanded by A^* search using our defined heuristic function. We will use the Chernoff bound to control the size of $\overline{R_\ell}$ which denotes the set of *non*-required nodes at depth ℓ.

Let v be a node at depth $\ell < (\epsilon_1+\epsilon_2)d$. Equation (4) implies

$$\Pr[\exists \text{ an overestimated node in } \textsc{Path}(v)] \le 2\ell b^{-\Delta} < 1/16\,.$$

The last inequality holds for sufficiently large d, as long as $\Delta = \text{poly}(d)$. On the other hand, if $\epsilon_1 < 1$, we have

$$\begin{aligned}
\Pr\left[v \in \overline{R_\ell}\right] &= \Pr\left[h^*(v) > \frac{(1+\epsilon_2)(d-\Delta)-\ell}{1-\epsilon_1}\right] \\
&\le \exp\left(-b^{\frac{(1+\epsilon_2)(d-\Delta)-\ell}{1-\epsilon_1}-d}\right) \qquad \text{(by Lemma 3.2)} \\
&= \exp\left(-b^{\frac{(\epsilon_1+\epsilon_2)d-(1+\epsilon_2)\Delta-\ell}{1-\epsilon_1}}\right)\,. \qquad (5)
\end{aligned}$$

Now set $\ell = (\epsilon_1+\epsilon_2)d - (1+\epsilon_2)\Delta - \log_b 4$. Then Equation (5) implies

$$\Pr\left[v \in \overline{R_\ell}\right] \le \exp\left(-b^{\frac{\log_d 4}{1-\epsilon_1}}\right) \le e^{-4} \le 1/16\,.$$

In the case $\epsilon_1 = 1$, the event "$(1-\epsilon_1)h^*(v) > (1+\epsilon_2)(d-\Delta) - \ell$" never happens given the value of ℓ that has been set. Hence, in any case, $\Pr\left[v \in \overline{R_\ell}\right] \le 1/8$ so that $\mathbb{E}\left[\left|\overline{R_\ell}\right|\right] \le b^\ell/8$. Applying the Chernoff bound in Corollary 3.1 again yields

$$\Pr\left[\left|\overline{R_\ell}\right| > eb^\ell/8\right] \le \exp(-b^\ell/8)\,.$$

Let E_{thin} be the event that $\left|\overline{R_\ell}\right| \ge b^\ell/2$. Since $b^\ell/2 > eb^\ell/8$,

$$\Pr[E_{\text{thin}}] \le \exp(-b^\ell/8)\,.$$

Putting the pieces together, we have

$$\Pr\left[A^* \text{ expands less than } b^\ell/2 \text{ nodes}\right] \le \Pr[E_{\text{close}} \vee E_{\text{thin}}] \le 2b^{-\Delta} + e^{-b^\ell/8}\,.$$

Setting $\Delta = 2\sqrt{d}$ we have $\ell = (\epsilon_1+\epsilon_2)d - 2(1+\epsilon_2)\sqrt{d} - \log_b 4$, and thus

$$\Pr\left[A^* \text{ expands less than } b^{(\epsilon_1+\epsilon_2)d-4\sqrt{d}}/8 \text{ nodes}\right] \le b^{-\sqrt{d}}$$

for sufficiently large d. $\square$

For contrast, we now explore the behavior of A^* with an adversarially selected solution set; this achieves a lower bound which is nearly tight (in comparison with the general upper bound on the worst-case running time of A^* obtained by setting $\gamma = 0$ in the bound of Lemma 3.1 above).

Theorem 4.2. *For any $d > 1$, there exists a solution set S whose optimal solutions lie at depth d and an (ϵ_1, ϵ_2)-approximate heuristic function h such that the A^* on the tree $\mathcal{T}$ using h expands at least $b^{(1+\epsilon_2)d-1-\epsilon_2/\epsilon_1}$ nodes.*

Proof. Consider a solution set S in which all ϵ_2-optimal solutions share an ancestor u lying at depth 1. Furthermore, S contains every node at depth $(1 + \epsilon_2)d$ that is *not a descendant of* u, where $d = h^*(r)$.

Now define an (ϵ_1, ϵ_2)-approximate heuristic h as follows: $h(u) = (1 + \epsilon_2)h^*(u)$ and $h(v) = (1-\epsilon_1)h^*(v)$ for all $v \neq u$. With this heuristic, every ϵ_2-optimal solution is "hidden" from the search procedure by its ancestor u. Precisely, since $f(u) = 1 + (1 + \epsilon_2)(d - 1) = (1 + \epsilon_2)d - \epsilon_2$, every ϵ_2-optimal solution s (which is a descendant of u) will have

$$\max_{v \in \textsc{Path}(s)} f(v) \geq f(u) = (1 + \epsilon_2)d - \epsilon_2 \,.$$

Thus $M \geq (1 + \epsilon_2)d - \epsilon_2$, where M is the min-max value defined in Equation (2).

Let v be any node at depth $\ell \leq (1+\epsilon_2)d$ that does not lie inside of $\textsc{SubTree}(u)$. Note that the f values monotonically increase along the path from root r to v, which implies that the node v must be expanded if $f(v) < M$. On the other hand, since every non-descendant of u at depth $(1+\epsilon_2)d$ is a solution, we have $\ell + h^*(v) \leq (1 + \epsilon_2)d$, and thus

$$f(v) \leq \ell + (1 - \epsilon_1)[(1 + \epsilon_2)d - \ell] = (1 - \epsilon_1)(1 + \epsilon_2)d + \epsilon_1\ell \,.$$

Hence, the node v must be expanded if $(1 - \epsilon_1)(1 + \epsilon_2)d + \epsilon_1\ell < (1 + \epsilon_2)d - \epsilon_2$, which is equivalent to $\ell < (1 + \epsilon_2)d - \epsilon_2/\epsilon_1$. It follows that the number of nodes expanded by A^* is at least

$$\sum_{\ell=0}^{(1+\epsilon_2)d-1-\epsilon_2/\epsilon_1} b^\ell - \sum_{\ell=0}^{(1+\epsilon_2)d-2-\epsilon_2/\epsilon_1} b^\ell = b^{(1+\epsilon_2)d-1-\epsilon_2/\epsilon_1} \,.$$

□

According to Theorem 4.2, if we set $\epsilon_2 = 0$ and let ϵ_1 be arbitrarily small provided $\epsilon_1 > 0$, then we can obtain a near-accurate heuristic which forces A^* to expand at least as many as b^{d-1} nodes. This lower bound partially explains why A^* can perform so poorly, even with an almost perfect heuristic, in certain applications (Helmert & Röger, 2008): The adversarially-chosen solution set given in the proof of Theorem 4.2 has an overwhelming number of near-optimal solutions. Indeed,

$$N_{\epsilon+\epsilon_2} \geq b^{(1+\epsilon_2)d} - b^{(1+\epsilon_2)d-1} \geq b^{(1+\epsilon_2)d-1}$$

for any $\epsilon > 0$.

5. Generalizations: Non-uniform Edge Costs and Branching Factors

In this section, we discuss how the generic upper bounds of Lemma 3.1 can be generalized to apply to more natural search models such as those with non-uniform branching factors and non-uniform edge costs; in Section 6, we show how these can be extended to general graph search models.

Now we consider a general search tree without the assumptions of uniform branching factor and uniform edge costs. From the same argument given in the proof of Lemma 3.1, we derive the assertion that when the heuristic is (ϵ_1, ϵ_2)-approximate, any node of cost more than $(\gamma\epsilon_1 + \epsilon_2 + 1 - \gamma)c^*$ will not be potentially expanded if it does not lie on a $(\gamma\epsilon_1 + \epsilon_2)$-optimal solution path, where γ is an arbitrary nonnegative number and $c^* = h^*(r)$ is the optimal solution cost.

Hence, the number of nodes potentially expanded by A^* with an (ϵ_1, ϵ_2)-approximate heuristic is bounded by

$$F\big((\gamma\epsilon_1 + \epsilon_2 + 1 - \gamma)c^*\big) + R\big((\gamma\epsilon_1 + \epsilon_2 + 1 - \gamma)c^*, \gamma\epsilon_1 + \epsilon_2\big) \,. \tag{6}$$

Here $F(\xi)$ is the number of nodes with cost no more than ξ, which we call *free* nodes; $R(\xi, \delta)$ is the number of nodes with cost in the range $(\xi, (1+\epsilon_2)c^*]$ that lie on a δ-optimal solution path, which we call *restricted* nodes.

To bound the number of free and restricted nodes, respectively, we assume that the branching factors are upper bounded and edge costs are lower bounded. Let $B \geq 2$ be the maximal branching factor and let m be the minimal edge cost. Since any node with cost no more than ξ must lie at depth no larger than ξ/m, we have

$$F(\xi) \leq 2B^{\xi/m}\,.$$

On each δ-optimal solution path, there are at most $((1+\epsilon_2)c^* - \xi)/m$ nodes of cost in the range $(\xi, (1+\epsilon_2)c^*]$. Thus,

$$R(\xi, \delta) \leq \frac{(1+\epsilon_2)c^* - \xi}{m} \times N_\delta\,.$$

Letting $\xi = (\gamma\epsilon_1 + \epsilon_2 + 1 - \gamma)c^*$, $\delta = \gamma\epsilon_1 + \epsilon_2$, and applying the bounds for $F(\xi)$ and $R(\xi, \delta)$ to the bound in (6), we obtain another upper bound on the number of expanded nodes when the heuristic is (ϵ_1, ϵ_2)-approximate:

$$2B^{(\gamma\epsilon_1+\epsilon_2+1-\gamma)c^*/m} + N_{\gamma\epsilon_1+\epsilon_2}(1-\epsilon_1)\gamma c^*/m \tag{7}$$

for any $\gamma \geq 0$. This equation (7) is a generalized version of the bound in Lemma 3.1. Substituting $\gamma = 1$ in (7), we arrive at the following simpler upper bound on the number of expanded nodes:

$$2B^{(\epsilon_1+\epsilon_2)c^*/m} + N_{\epsilon_1+\epsilon_2}(1-\epsilon_1)c^*/m\,. \tag{8}$$

6. Bounding Running Time of A^* on Graphs

In previous parts, we have established bounds on the running time of A^* on the tree model. Now we will apply those bounds to A^* on the graph model. In order to do that, we will first unroll the graph into a cover tree, and then bound the number of solutions lifted to the cover tree.

6.1 Unrolling Graphs into Trees

The preceding generic upper bounds are developed for tree-based models; in this section we discuss a natural extension to general graph search models. The principal connection is obtained by "unrolling" a graph into a tree on which A^* expands at least as many nodes as it does on the original graph (including repetitions). More specifically, given a directed graph G and starting node x_0 in G, we define a *cover tree* $\mathcal{T}(G)$ whose nodes are in one-to-one correspondence with finite-length paths in G from x_0. We shall write a path $(x_0, \ldots, x_\ell)$ in G as a node in $\mathcal{T}(G)$. The root of $\mathcal{T}(G)$ is (x_0). The parent of a node $(x_0, x_1, \ldots, x_\ell)$ in $\mathcal{T}(G)$ is the node $(x_0, x_1, \ldots, x_{\ell-1})$, and the edge cost between the two nodes $(x_0, x_1, \ldots, x_{\ell-1})$ and $(x_0, x_1, \ldots, x_\ell)$ in $\mathcal{T}(G)$ equals the cost of the edge $(x_{\ell-1}, x_\ell)$ in G. Hence, for each node P in $\mathcal{T}(G)$, the cost value $g(P)$ is equal to the total edge cost on the path P in G. A node $(x_0, \ldots, x_\ell)$ in $\mathcal{T}(G)$ is designated as a solution whenever x_ℓ is a solution in G.

A node in $\mathcal{T}(G)$ that corresponds to a path ending at node $x \in G$ will be called a *copy* of x. Observe that a solution in G may "lift" multiple times to solutions in $\mathcal{T}(G)$, as each node in G may have multiple copies in $\mathcal{T}(G)$. Figure 1 illustrates an example of unrolling a graph into a cover tree. In this example, node s is a solution in the graph and its first two copies in the cover tree correspond to the paths $(0, 3, s)$ and $(0, 5, 3, s)$, where 0 is the starting node in the given graph.

The A^* search on graph G is described in Algorithm 2 below, in which $h(x)$ is the heuristic at node x, $g(x)$ is the cost of the *current* path from x_0 to x, and $c(x, x')$ denotes the cost of the edge (x, x') in G. We assume the value of $h(x)$ depends only on x, i.e., $h(x)$ does not depend on a particular path from x_0 to x. Unlike A^* search on a tree, for each node x in Open or Closed,

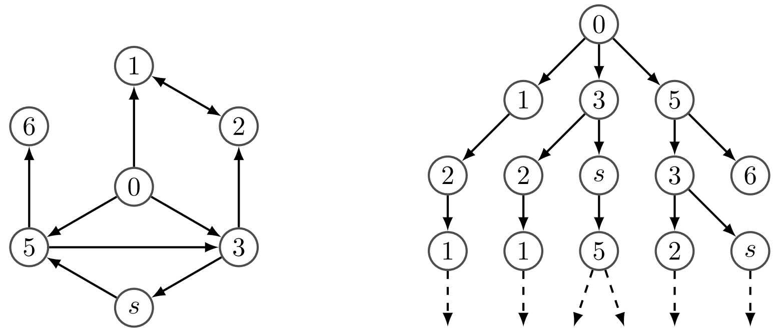

Figure 1: Unrolling a graph into a cover tree.

Algorithm 2 also keeps track of the current path P from x_0 to x through the pointers, and the current f-value of x is equal to $g(P) + h(x)$. This current path is the cheapest path from x_0 to x that passes only nodes that have been expanded.

Algorithm 2 A^* search on a graph (Pearl, 1984, p. 64)

1. Initialize Open $:= \{x_0\}$ and $g(x_0) := 0$.
2. Repeat until Open is empty.
   (a) Remove from Open and place on Closed a node x for which the function $f = g + h$ is minimum.
   (b) If x is a solution, exit with success and return x.
   (c) Otherwise, expand x, generating all its successors. For each successor x' of x,
      i. If x' is not on Open or Closed, estimate $h(x')$ and calculate $f(x') = g(x') + h(x')$ where $g(x') = g(x) + c(x, x')$, and put x' to Open with pointer back to x.
      ii. If x' is on Open or Closed, compare $g(x')$ and $g(x) + c(x, x')$. If $g(x) + c(x, x') < g(x')$, direct the pointer of x' back to x and reopen x' if it is in Closed.
3. Exit with failure.

Now consider A^* search on the cover tree $\mathcal{T}(G)$ of graph G using the same heuristic function h: for each node P in $\mathcal{T}(G)$, set the heuristic value $h(P)$ to be equal to $h(x)$ if P is a copy of node $x \in G$, i.e., P is a path in G from x_0 to x. Observe that the cover tree $\mathcal{T}(G)$ and the graph G share the same threshold M (defined in Equation (2)). Hence, whenever a node $x \in G$ is expanded with current path P, we must have $g(P) + h(x) \le M$, which implies that P is potentially expanded by A^* search on the cover tree $\mathcal{T}(G)$. This shows the following fact:

Fact 3. *The number of node expansions by A^* on G is no more than the number of nodes potentially expanded by A^* on $\mathcal{T}(G)$ using the same heuristic.*

Here, by *node expansion*, we mean an execution of the expand step of A^*, i.e. Step (2c). Note that, in general, a node in G can be expanded many times along different paths.

Remark The running time of A^* on the cover tree can also be used to upper bound the running time of iterative-deepening A^* (IDA*) on the graph. Recall that the running time of IDA* is dominated by its last iteration. On the other hand, the last iteration of IDA* on G is merely depth-first search on the cover tree $\mathcal{T}(G)$ up to the cost threshold M. Hence, the number of expansions in the last iteration of IDA* is no more than the number of nodes potentially expanded by A^* on the cover tree.

So, to upper-bound time complexity of A^* or IDA* on a graph, it suffices to unroll the graph into the cover tree and apply upper bounds on the number of nodes potentially expanded by A^* on the cover tree. In particular, the bound in Equation (7) can be applied directly to the A^* search on G.

Note that while these bounds can be applied directly, the problem of determining exactly how solutions in G lift to solutions in the cover tree depends on delicate structural properties of G—specifically, it depends on the growth of the *number of distinct paths from x_0 to a solution* as a function of the length of these paths. In particular, in order to obtain general results on the complexity of A^* in this model, we must invoke some measure of the connectedness of the graph G. Below we show how to bound the complexity of A^* in terms of spectral properties of G. We choose this approach because it offers a single parameter notion of connectedness (the second eigenvalue) that is both analytically tractable and can actually be analyzed or bounded for many graphs of interest, including various families of Cayley graphs and combinatorial graphs by methods such as conductance.

6.2 An Upper Bound via Graph Spectra

We shall consider an undirected[2] graph G on n vertices as the search space. Let x_0 be the starting node and let S be the set of solutions in G. For simplicity, assume G is b-regular ($2 < b \ll n$) and the edge costs are uniformly equal to one, so the cover tree $\mathcal{T}(G)$ is b-ary and has uniform edge cost. We assume, additionally, that $|S|$ is treated as a constant when $n \to \infty$.

By Fact 3 and Lemma 3.1, the number of node expansions by A^* on G with an (ϵ_1, ϵ_2)-approximate heuristic is at most $2b^{(\epsilon_1+\epsilon_2)d} + dN_{\epsilon_1+\epsilon_2}$, where d is the optimal solution cost, which equals the optimal solution depth in $\mathcal{T}(G)$, and N_δ is the number of δ-optimal solutions in $\mathcal{T}(G)$. Our goal now is to upper bound N_δ (of the cover tree $\mathcal{T}(G)$) in terms of spectral properties of G.

We introduce the principal definitions of spectral graph theory below, primarily to set down notation. A more complete treatment of spectral graph theory can be found in the work of Chung (1997).

Graph Spectra. For a graph G, let A be the adjacency matrix of G: $A(x, y) = 1$ if x is adjacent to y, and 0 otherwise. This is a real, symmetric matrix ($A^{\mathsf{T}} = A$) and thus has real eigenvalues $b = \mu_1 \geq \mu_2 \geq \ldots \geq \mu_n \geq -b$, by the spectral theorem (Horn & Johnson, 1999). Let $\widehat{A} = {}^1\!/\!_b \cdot A$ denote the normalized adjacency matrix of G; then $\widehat{A}$ has eigenvalues $1 = \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq -1$, which are referred to as the *spectrum* of G, where $\lambda_i = \mu_i/b$. These eigenvalues, along with their associated eigenvectors, determine many combinatorial aspects of the graph G. In most applications of graph eigenvalues, however, only the critical value $\mu = \mu(G) \stackrel{\text{def}}{=} \max\{|\mu_2|, |\mu_n|\}$ is invoked and, moreover, the real parameter of interest is the *gap* between $\lambda = \mu/b$ and the largest eigenvalue $\lambda_1 = 1$ of the normalized adjacency matrix. Intuitively, λ measures the "connectedness" of G. Sparsely connected graphs have $\lambda \approx 1$; for the n-cycle, for example, $\lambda = 1 - O(1/n)$. The hypercube on $N = 2^n$ vertices has $\lambda = 1 - \Theta(1/\log N)$. Similar bounds on μ and λ, are known for many families of Cayley graphs. Random graphs, even of constant degree $b \geq 3$, achieve $\lambda = o(1)$ with high probability. In fact, a recent result of Friedman (2003) strengthens this:

Theorem 6.1. *(Friedman, 2003) Fix a real $c > 0$ and an integer $b \geq 2$. Then with probability $1 - o(1)$ (as $n \to \infty$),*

$$\mu(G_{n,b}) \leq 2\sqrt{b-1} + c$$

2. While one can produce an analogous cover tree in the directed case, the spectral machinery we apply in the next section is somewhat complicated by the presence of directed edges. See the work of Chung (2006) and Horn and Johnson (1999, Perron-Frobenius theorem) for details.

where $G_{n,b}$ is a random b-regular graph on n vertices.

We remark that for any non-bipartite connected graph with diameter D, we always have $\mu \leq b - 1/(Dn)$. Under stronger conditions, when the graph is vertex-transitive (which is to say that for any pair v_0, v_1 of vertices of G there is an automorphism of G sending v_0 to v_1), one has $\mu \leq b - \Omega(1/D^2)$ (Babai, 1991). While vertex transitivity is a strong condition, it is satisfied by many natural algebraic search problems (e.g., 15-puzzle-like search spaces and the Rubik's cube).

The principal spectral tool we apply in this section is described in Lemma 6.1 below. We begin with some notation.

Notations. Any function ϕ on G can be viewed as a column vector indexed by the vertices in G and vice versa. For each vertex $x \in G$, let $\mathbf{1}_x$ denote the function on G that has value 1 at x and 0 at every vertex other than x. For any real-valued functions ϕ, φ on G, define the inner product $\langle \phi, \varphi \rangle = \sum_{x \in G} \phi(x)\varphi(x)$. We shall use $\| \cdot \|$ to denote the L_2-norm, i.e., $\|\phi\| = \sqrt{\langle \phi, \phi \rangle}$ for any function ϕ on G.

Recall that since $\widehat{A}$ is symmetric and real, by spectral theorem (Horn & Johnson, 1999), there exist associated eigenfunctions $\phi_1, \ldots, \phi_n$ that form an orthonormal basis for the space of real-valued functions on G, where ϕ_i is the eigenfunction associated with the eigenvalue λ_i of $\widehat{A}$. In particular, we have $\widehat{A}\phi_i = \lambda_i \phi_i$ and $\|\phi_i\| = 1$ for all i, and $\langle \phi_i, \phi_j \rangle = 0$ for all $i \neq j$. In this basis, we can write $\phi = \sum_{i=1}^n \langle \phi, \phi_i \rangle \phi_i$ for any real-valued function ϕ on G.

Lemma 6.1. *Let G be an undirected b-regular graph with n vertices, and $\lambda = \mu(G)/b$. For any probability distributions p and q on vertices of G, and any integers $s, t \geq 0$,*

$$\left| \left\langle \widehat{A}^s p, \widehat{A}^t q \right\rangle - \frac{1}{n} \right| \leq \lambda^{s+t} \left(\|p\| \cdot \|q\| - \frac{1}{n} \right) .$$

Proof. Write $p = \sum_{i=1}^n a_i \phi_i$ and $q = \sum_{j=1}^n b_j \phi_j$ where $a_i = \langle p, \phi_i \rangle, b_j = \langle q, \phi_j \rangle$. Then

$$\left\langle \widehat{A}^s p, \widehat{A}^t q \right\rangle = \left\langle \sum_{i=1}^n a_i \lambda_i^s \phi_i, \sum_{j=1}^n b_j \lambda_j^t \phi_j \right\rangle = \sum_{i,j=1}^n a_i b_j \lambda_i^s \lambda_j^t \langle \phi_i, \phi_j \rangle = \sum_{i=1}^n \lambda_i^{s+t} a_i b_i \, .$$

By the Cauchy-Schwartz inequality,

$$\sum_{i=1}^n |a_i b_i| \leq \sqrt{\left(\sum_{i=1}^n a_i^2 \right) \left(\sum_{i=1}^n b_i^2 \right)} = \|p\| \cdot \|q\| \, .$$

Without loss of generality, assume $\phi_1(x) = 1/\sqrt{n}$ for all vertices $x \in G$. Since p is a probability distribution,

$$a_1 = \langle p, \phi_1 \rangle = \sum_{x \in G} p(x)\phi_1(x) = \frac{1}{\sqrt{n}} \sum_{x \in G} p(x) = \frac{1}{\sqrt{n}} \, .$$

Similarly, $b_1 = \frac{1}{\sqrt{n}}$. Thus, $a_1 b_1 = \frac{1}{n}$. So we have

$$\begin{aligned} \left| \left\langle \widehat{A}^s p, \widehat{A}^t q \right\rangle - \frac{1}{n} \right| &= \left| \sum_{i=2}^n \lambda_i^{s+t} a_i b_i \right| \\ &\leq \lambda^{s+t} \sum_{i=2}^n |a_i b_i| \qquad (\text{as } \lambda = \max_{2 \leq i \leq n} |\lambda_i|) \\ &\leq \lambda^{s+t} \left(\|p\| \cdot \|q\| - \frac{1}{n} \right) , \end{aligned}$$

completing the proof of the lemma. □

With Lemma 6.1 in hand, we establish the following bound on the number of paths of a prescribed length ℓ connecting a pair of vertices. We then apply this to control the number of ϵ-optimal solutions in the cover tree of G. Let $P_\ell(u, v)$ denote the number of paths in G of length ℓ from u to v.

Lemma 6.2. *Let G be an undirected b-regular graph with n vertices, and $\mu = \mu(G)$. For any vertices u, v in G and $\ell \geq 0$,*

$$\left| P_\ell(u,v) - \frac{b^\ell}{n} \right| \leq \mu^\ell \left(1 - \frac{1}{n}\right) < \mu^\ell\,.$$

Proof. Since $P_\ell(u, v)$ is the number of ℓ-length paths from u to v, we have $P_\ell(u, v) = b^\ell p^{(\ell)}(v)$, where $p^{(\ell)}(v)$ is the probability that a natural random walk on G of length ℓ starting from u ends up at v. Since $p^{(\ell)} = \widehat{A}^\ell \mathbf{1}_u$ and $p^{(\ell)}(v) = \langle \mathbf{1}_v, p^{(\ell)} \rangle$, we have $\frac{P_\ell(u,v)}{b^\ell} = \left\langle \mathbf{1}_v, \widehat{A}^\ell \mathbf{1}_u \right\rangle$. Applying Lemma 6.1 yields

$$\left| \frac{P_\ell(u,v)}{b^\ell} - \frac{1}{n} \right| = \left| \left\langle \mathbf{1}_v, \widehat{A}^\ell \mathbf{1}_u \right\rangle - \frac{1}{n} \right| \leq \lambda^\ell \left(\|\mathbf{1}_v\| \cdot \|\mathbf{1}_u\| - \frac{1}{n} \right) = \lambda^\ell \left(1 - \frac{1}{n}\right).$$

As $\lambda = \mu/b$, multiplying both sides of the last inequality by b^ℓ completes the proof for the lemma. □

The major consequence of Lemma 6.2 in our application is the following bound on the number of ϵ-optimal solutions in $\mathcal{T}(G)$.

Theorem 6.2. *Let G be an undirected b-regular graph with n vertices, and $\mu = \mu(G)$. For sufficiently large n and any $\epsilon \geq 0$, the number of ϵ-optimal solutions in $\mathcal{T}(G)$ is*

$$N_\epsilon < 2|S| \left(\frac{b^{(1+\epsilon)d}}{n} + \mu^{(1+\epsilon)d} \right),$$

where d is the depth of optimal solutions in $\mathcal{T}(G)$, and S is the set of solution nodes in G.

Proof. For each solution $s \in S$, the number of copies of s at level ℓ in $\mathcal{T}(G)$ equals $P_\ell(x_0, s)$, which is less than $b^\ell/n + \mu^\ell$ by Lemma 6.2. Hence, the number of solutions at level ℓ in $\mathcal{T}(G)$ is

$$\sum_{s \in S} P_\ell(x_0, s) < |S| \left(\frac{b^\ell}{n} + \mu^\ell \right). \tag{9}$$

Summing up both sides of (9) for ℓ ranging from d to $(1 + \epsilon)d$, we have

$$N_\epsilon = \sum_{\ell=d}^{(1+\epsilon)d} \sum_{s \in S} P_\ell(x_0, s) < |S| \left(\frac{1}{n} \sum_{\ell=d}^{(1+\epsilon)d} b^\ell + \sum_{\ell=d}^{(1+\epsilon)d} \mu^\ell \right).$$

When n is sufficiently large, we have $\mu \geq 2$. Thus,

$$N_\epsilon < |S| \left(\frac{1}{n} 2 b^{(1+\epsilon)d} + 2\mu^{(1+\epsilon)d} \right).$$

□

Note that $b^{(1+\epsilon)d}/n = O(1)$ if $d = O(\log_b n)$. As mentioned earlier (Theorem 6.1), most b-regular graphs have $\mu \leq 2\sqrt{b-1} + o(1) \leq 2\sqrt{b}$. Assuming G has this spectral property and $d = O(\log_b n)$, Theorem 6.2 gives

$$N_\epsilon = O(\mu^{(1+\epsilon)d}) = O\left(2^{(1+\epsilon)d} b^{(1+\epsilon)d/2}\right).$$

In such cases, the number of node expansions by A^* on G using an $(\epsilon_1, \epsilon - \epsilon_1)$-approximate heuristic is $O(d 2^{(1+\epsilon)d} b^{(1+\epsilon)d/2})$, which implies the effective branching factor of A^* is roughly bounded by $2^{1+\epsilon} b^{(1+\epsilon)/2} < 8 b^{(1+\epsilon)/2}$.

7. Experimental Results

As discussed in the introduction, the bounds established thus far guarantee that E, the number of nodes expanded by A^* using a δ-accurate heuristic, satisfies

$$E \leq 2b^{\delta d} + dN_\delta \approx cb^{\delta d}$$

under the assumption that $N_\delta \approx b^{\delta d}$. (Here, as before, b is the branching factor, d is the optimal solution depth, and c is some constant.) This suggests the hypothesis that for hard combinatorial problems with suitably sparse near-optimal solutions,

$$\log E \approx \delta d \log b + \xi\,. \tag{10}$$

where ξ is a constant determined by the search space and heuristic but independent from δ. In particular, this suggests a linear dependence of $\log E$ on δ. We experimentally investigated this hypothesized relationship with a family of results involving the *Knapsack* problem and the *partial Latin square* problem. As far as we are aware, these are the first experimental results specifically investigating this dependence.

We remark that in order for such an experimental framework to really cast light on the bounds we have presented for A^*, one must be able to furnish a heuristic with *known approximation guarantees.*

7.1 A^* Search for Knapsack

We begin with describing a family of experimental results for A^* search coupled with approximate heuristics for solving the Knapsack problem. This problem has been extremely well-studied by a wide variety of fields including finance, operations research, and cryptography (Kellerer, Pferschy, & Pisinger, 2004). As the Knapsack problem is NP-hard (Karp, 1972), no efficient algorithm can solve it exactly unless NP = P. Despite that, this problem admits an FPTAS (Vazirani, 2001, p. 70), an algorithm that will return an ϵ-approximation to the optimal solution in time polynomial in both $1/\epsilon$ and the input size. We use this FPTAS to construct approximate *admissible* heuristics for the A^* search, which yields an *exact* algorithm for Knapsack that may expand far fewer nodes than straightforward exhaustive search. (Indeed, the resulting algorithm is, in general, more efficient than exhaustive search.)

7.1.1 A Search Model for Knapsack

Consider a Knapsack instance given by n items, and let $[n] = \{1, \ldots, n\}$. Each item $i \in [n]$ has weight $w_i > 0$ and profit $p_i > 0$. The knapsack has capacity $c > 0$. The task is to find a set of items with maximal total profit such that its total weight is at most c. This Knapsack instance will be denoted as a tuple $\langle [n], p, w, c\rangle$. The Knapsack instance restricted to a subset $X \subset [n]$ is denoted $\langle X, p, w, c\rangle$. For each subset $X \subset [n]$, we will let $w(X)$ and $p(X)$ denote the total weight and the total profit, respectively, of all items in X, i.e., $w(X) = \sum_{i \in X} w_i$ and $p(X) = \sum_{i \in X} p_i$.

Search Space. We represent the Knapsack instance $\langle [n], p, w, c\rangle$ as a search space as follows. Each state (or node) in the search space is a nonempty subset $X \subset [n]$. A move (or edge) from one state X to another state is taken by removing an item from X. The cost of such a move is the profit of the removed item. A state $X \subset [n]$ is designated as a solution if $w(X) \leq c$. The initial state is the set $[n]$. See Figure 2 for an example of the search space with $n = 4$.

This search space is an irregular directed graph whose out-degrees span in a wide range, from 2 to $n - 1$. Moreover, for any two states X_1, X_2 with $X_2 \subset X_1 \subset [n]$, there are $|X_1 \setminus X_2|!$ paths on this search graph from X_1 to X_2. Moreover, every path from X_1 to X_2 has the same cost equal to $p(X_1) - p(X_2)$. This feature of the search graph makes A^* behave like it does on a spanning subtree of the graph: no state in this search graph will be reopened. Hence, for any state $X \subset [n]$, the cost

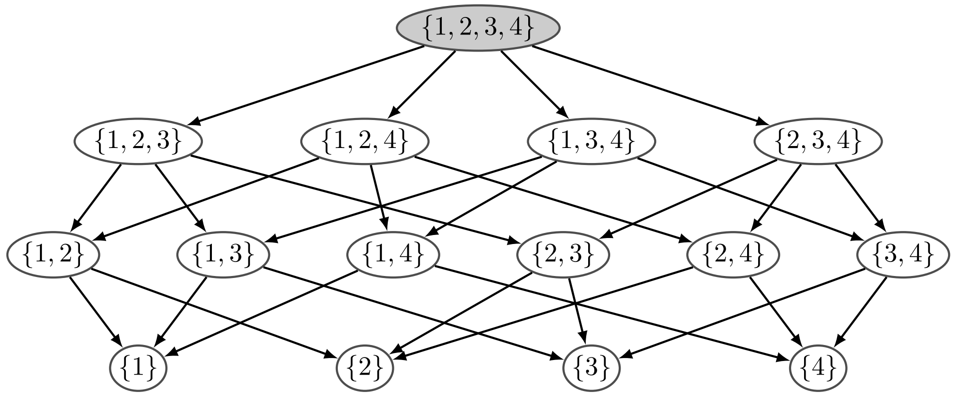

Figure 2: The search space for a Knapsack instance given by the set of 4 items $\{1, 2, 3, 4\}$. Solution states and edge costs are not indicated in this figure.

of any path from the starting state to X is

$$g(X) = p([n] \setminus X) = p([n]) - p(X)\,,$$

and the cheapest cost to reach a solution from a state $X \subset [n]$ is

$$h^*(X) = p(X) - \mathrm{Opt}(X)\,,$$

where $\mathrm{Opt}(X)$ is the total profit of an optimal solution to the Knapsack instance $\langle X, p, w, c\rangle$, i.e.,

$$\mathrm{Opt}(X) \overset{\text{def}}{=} \max\{p(X') \mid X' \subset X \text{ and } w(X') \leq c\}\,.$$

Observe that a solution state $X^* \subset [n]$ on the search space $\langle [n], p, w, c\rangle$ is optimal if and only if $g(X^*)$ is minimal, or equivalently, $p(X^*)$ is maximal, which means that X^* is an optimal solution to the Knapsack instance $\langle [n], p, w, c\rangle$.

Heuristic Construction. Fix a constant $\delta \in (0, 1)$. In order to prove the linear dependence of $\log E$ on δ, we wish to have an efficient δ-accurate heuristic H_δ on the aforementioned Knapsack search space $\langle [n], p, w, c\rangle$. Moreover, in order to guarantee that the solution returned by the A^* search is optimal, we insist that H_δ be admissible, so H_δ must satisfy:

$$(1-\delta)h^*(X) \leq H_\delta(X) \leq h^*(X) \quad \forall X \subset [n]\,.$$

The main ingredient for constructing such a heuristic is an FPTAS described in the book of Vazirani (2001, p. 70). This FPTAS is an algorithm, denoted $\mathcal{A}$, that returns a solution with total profit at least $(1-\epsilon)\mathrm{Opt}(X)$ to each Knapsack instance $\langle X, p, w, c\rangle$ and runs in time $O\left(|X|^3/\epsilon\right)$, for any $\epsilon \in (0, 1)$. For each nonempty subset $X \subset [n]$, let $\mathcal{A}_\epsilon(X)$ denote the total profit of the solution returned by algorithm $\mathcal{A}$ with error parameter ϵ to the Knapsack instance $\langle X, p, w, c\rangle$. Then we have for any $\epsilon \in (0, 1)$,

$$(1-\epsilon)\mathrm{Opt}(X) \leq \mathcal{A}_\epsilon(X) \leq \mathrm{Opt}(X)\,,$$

which implies

$$p(X) - \frac{\mathcal{A}_\epsilon(X)}{1-\epsilon} \leq h^*(X) \leq p(X) - \mathcal{A}_\epsilon(X)\,. \tag{11}$$

Thus we may work with the heuristic $H_\delta(X) = p(X) - \frac{\mathcal{A}_\epsilon(X)}{1-\epsilon}$, which guarantees admissibility. However, this definition of H_δ does not guarantee δ-approximation for H_δ: with this definition, the condition $(1-\delta)h^*(X) \leq H_\delta(X)$ is equivalent to

$$(1-\delta)h^*(X) \leq p(X) - \frac{\mathcal{A}_\epsilon(X)}{1-\epsilon}\,, \tag{12}$$

which does not always hold. Since $h^*(X) \le p(X) - \mathcal{A}_\epsilon(X)$, the condition of (12) will be satisfied if

$$(1-\delta)(p(X) - \mathcal{A}_\epsilon(X)) \le p(X) - \frac{\mathcal{A}_\epsilon(X)}{1-\epsilon}\,. \tag{13}$$

Hence, we will define $H_\delta(X) = p(X) - \frac{\mathcal{A}_\epsilon(X)}{1-\epsilon}$ if Equation (13) holds. Otherwise, we will define $H_\delta(X)$ differently, still ensuring that it is δ-approximate and admissible. Note that if X is not a solution, at least one item in X must be removed in order to reach a solution contained in X, thus $h^*(X) = p(X) - \mathrm{Opt}(X) \ge m$, where m is the smallest profit of all items. This gives another option to define $H_\delta(X)$ that will guarantee the admissibility. In summary, we define the heuristic function H_δ as follows: for all non-solution state X,

$$H_\delta(X) \overset{\text{def}}{=} \begin{cases} p(X) - \frac{\mathcal{A}_\epsilon(X)}{1-\epsilon} & \text{if (13) holds} \\ m & \text{otherwise,} \end{cases} \tag{14}$$

where ϵ will be determined later so that H_δ is δ-approximate. If X is a solution, we simply set $H_\delta(X) = 0$, because $h^*(X) = 0$ in this case. Then H_δ is admissible, regardless of ϵ.

To make sure that H_δ is δ-approximate, it remains to consider the case when (13) does not hold, i.e., $p(X) - \frac{\mathcal{A}_\epsilon(X)}{1-\epsilon} < (1-\delta)(p(X) - \mathcal{A}_\epsilon(X))$, for any non-solution state X. In such a case, we have

$$p(X) - \mathcal{A}_\epsilon(X) \le \frac{\epsilon}{(1-\epsilon)\delta}\mathcal{A}_\epsilon(X) \le \frac{\epsilon}{(1-\epsilon)\delta}(p([n]) - m)\,. \tag{15}$$

The last inequality is due to the assumption that X is not a solution. Now we want to choose ϵ such that

$$\frac{\epsilon}{(1-\epsilon)\delta}(p([n]) - m) \le \frac{m}{1-\delta} \tag{16}$$

which, combining with (11) and (15), will imply $(1-\delta)h^*(X) \le m = H_\delta(X)$. Therefore, we will choose ϵ such that

$$\epsilon^{-1} = 1 + \left(\delta^{-1} - 1\right)\left(p([n])/m - 1\right).$$

Since the running time to compute $\mathcal{A}_\epsilon(X)$ is $O\left(|X|^3\epsilon^{-1}\right)$, the running time to compute $H_\delta(X)$ will be $O\left(|X|^3\delta^{-1}p([n])/m\right)$, which is polynomial in both n and δ^{-1} if all the profits are bounded some range $[m, \mathrm{poly}(n)m]$. The A^* search using the heuristic H_δ for the given Knapsack space $\langle [n], p, w, c\rangle$ is described in Algorithm 3 below.

7.1.2 Experiments

In order to avoid easy instances, we focus on two families of Knapsack instances identified and studied by Pisinger (2005) that are difficult for existing exact algorithms, including dynamic programming algorithms and branch-and-bound algorithms:

Strongly Correlated: For each item $i \in [n]$, choose its weight w_i as a random integer in the range $[1, R]$ and set its profit $p_i = w_i + R/10$. This correlation between weights and profits reflects a real-life situation where the profit of an item is proportional to its weight plus some fixed charge.

Subset Sum: For each item $i \in [n]$, choose its weight w_i as a random integer in the range $[1, R]$ and set its profit $p_i = w_i$. Knapsack instances of this type are instances of the subset sum problem.

For our tests we set the data range parameter $R := 1000$ and choose the knapsack capacity as $c = (t/101)\sum_{i\in[n]} w_i$, where t is a random[3] integer in the range $[30, 70]$.

3. In the paper of Pisinger (2005), t is a fixed integer between 1 and 100, and the average runtime of all tests corresponding to all values of t was reported.

Algorithm 3 A^* Search for Knapsack

Input: $\langle n, p, w, c, \delta \rangle$; where n is the number of items, p_i and w_i are the profit and weight of item $i \in [n]$, c is the capacity of the knapsack, and $\delta \in (0, 1)$ is an error parameter for the heuristic.
Oracle: The FPTAS algorithm $\mathcal{A}$ for the Knapsack problem described by Vazirani (2001, p. 70).
Notation: For each subset $X \subset [n]$ of items, let $p(X) = \sum_{i \in X} p_i$, $w(X) = \sum_{i \in X} w_i$.
Output: a subset $X^* \subset [n]$ of items such that $w(X^*) \leq c$ and $p(X^*)$ is maximal.

1. Put the start node $[n]$ on Open. Let $m = \min_{1 \leq i \leq n} p_i$. Set ϵ such that

$$\epsilon^{-1} = 1 + \left(\delta^{-1} - 1\right) \left(p([n])/m - 1\right).$$

2. Repeat until Open is empty:
   (a) Remove from Open and place on Closed a node X for which $g(X) + h(X)$ is minimum.
   (b) If $w(X) \leq c$, exit with success and return X, an optimal solution.
   (c) Otherwise, expand X: For each item $i \in X$, let $X' = X \setminus \{i\}$,
      i. If X' is not on Open or Closed, set $g(X') := g(X) + p(i) = p([n]) - p(X')$, and compute the heuristic $h(X')$ as follows:
         A. If X' is a solution, set $h(X') := 0$.
         B. Otherwise, run algorithm $\mathcal{A}$ on the Knapsack input $\langle X', p, w, c \rangle$ with error parameter ϵ, and let $\mathcal{A}(X')$ denote the total profit of the solution returned by algorithm $\mathcal{A}$. Then set

$$h(X') := \begin{cases} p(X') - \frac{\mathcal{A}(X')}{1-\epsilon} & \text{if } p(X') - \frac{\mathcal{A}(X')}{1-\epsilon} \geq (1-\delta)(p(X') - \mathcal{A}(X')) \\ m & \text{otherwise.} \end{cases}$$

      Then put X' to Open with pointer back to X.
      ii. Otherwise (X' is on Open or Closed, so $g(X')$ has been calculated), if $g(X) + p(i) < g(X')$, direct the pointer of X' back to X and reopen X' if it is in Closed.
      [**Remark:** Since all paths from the starting node to X' have the same cost, the condition $g(X) + p(i) < g(X')$ never holds. In fact, this step can be discarded.]

3. Exit with failure.

After generating a Knapsack instance $\langle [n], p, w, c\rangle$ of either type described above, we run a series of the A^* search using the given heuristic H_δ, with various values of δ, as well as breath first search (BFS), to solve the Knapsack instance. When each search finishes, the values of E and d are reported, where E is the number of nodes (states) expanded by the search, and d is the depth of the optimal solution found by the search. In this Knapsack search space, k equals the number of items removed from the original set $[n]$ to obtain the optimal solution found by the search. The overall runtime for each search, including the time for computing the heuristic, is also reported. In addition, we report the optimal value $h^*([n])$ and the minimal edge cost m (i.e., minimal profit) of the search space for each Knapsack instance tested.

To specify appropriate size n for each Knapsack instance type, we ran a few exploratory experiments and identified the largest possible value of n for which most search instances would finish within a few hours. Then we chose those values of n ($n = 23$ for the Strongly Correlated type, and $n = 20$ for the Subset Sum type) for our final experiments. Observing that the optimal solution depths resulted from Knapsack instances of these sizes are fairly small, ranging from 5 to 15, we selected sample points for δ in the high interval $[0.5, 1)$ with a distance between two consecutive points large enough so that the sensitiveness of E to δ can be seen. In particular, we selected eight sample points for δ from $8/16 = 0.5$ to $15/16 = 0.9375$ with the distance of $1/16 = 0.0625$ between two consecutive points. In our final experiments, we generated 20 Knapsack instances of each type with the selected parameters for n and δ.

Experimental Results. Results for our final experiments are shown in Tables 1, 2, 3, 4, 5, and 6, in which the rows corresponding to breath first search are indicated with "BFS" under the column of δ. These data show, as expected, that A^* search outperforms breath first search in terms of the number of nodes expanded and, naturally, that the smaller δ, the fewer nodes A^* expands. As a result, the effective branching factor of A^* will decrease as δ decreases (as long as all optimal solutions in the given search space are located at the same depth). Recall that if A^* expands E nodes and finds a solution at depth d, then its *effective branching factor* is the branching factor of a uniform tree of depth d and E nodes (Russell & Norvig, 1995, p. 102), i.e., the number b^* satisfying $E = 1 + b^* + (b^*)^2 + \cdots + (b^*)^d$. Clearly, $(b^*)^d \leq E$ and, if $b^* \geq 2$, we have $E \leq 2(b^*)^d$. As we shall focus solely on values of $b^* \geq 2$, we simply use $E^{1/d}$ as a proxy for effective branching factor, content that this differs from the actually quantity by a factor no more than $2^{1/d}$. (Of course, as b^* grows this error decays even further). The effective branching factors, calculated as $E^{1/d}$, of A^* search and breath first search for Knapsack instances of type Strongly Correlated are shown in Tables 1, 2, and 3. Note that for Knapsack instances of the Subset Sum type, one cannot directly compare effective branching factors, as the optimal solutions found by different search instances can appear at different depths.

Our primary goal in these experiments is to investigate the proposed linear dependence which, in this case of non-uniform branching factors and non-uniform edge costs, we may express

$$\log E \approx \delta \overline{d} \log b_{\mathrm{BFS}} + \xi \,, \tag{17}$$

where $\overline{d}$ is the average optimal solution depth, b_{BFS} is the effective branching factor of breath first search, and ξ is a constant not depending on δ. To examine to what extend our data supports this hypothesis, we calculate the least-squares linear fit (or "linear fit" for short) of $\log E$ (for each Knapsack instance, varying δ) using the least-squares linear regression model, and measure the coefficient of determination R^2. In our experiments, 17 out of 20 Knapsack instances of type Strongly Correlated and all 20 Knapsack instances of type Subset Sum have the R^2 value at least 0.9. For these instances, over 90% of the variation in $\log E$ depends linearly on δ, a remarkable fit. See Figure 5 for detailed histograms of R^2 values for our Knapsack instances. The median R^2 is 0.9534 for Knapsack instances of type Strongly Correlated, and is 0.9797 for those of type Subset Sum. Graphs of $\log E$ and its linear fit for Knapsack instances with the median R^2 among those of the same type are shown in Figures 3 and 4. Note that as there are an even number of instances of each type, there is no single instance with the median value. The instances shown in these graphs actually have the R^2 value below the median.

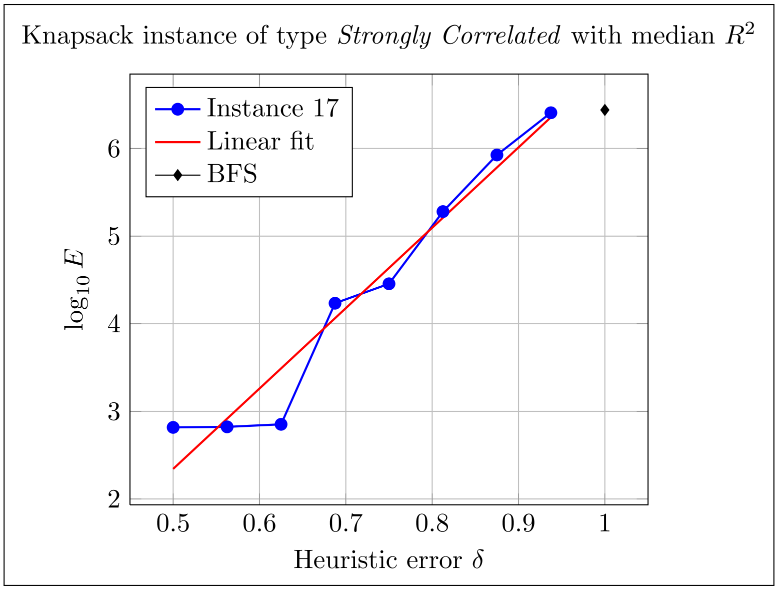

Figure 3: Graph of $\log_{10} E$ and its least-squares linear fit for the Knapsack instance of type Strongly Correlated with the median R^2 (see data in Table 3).

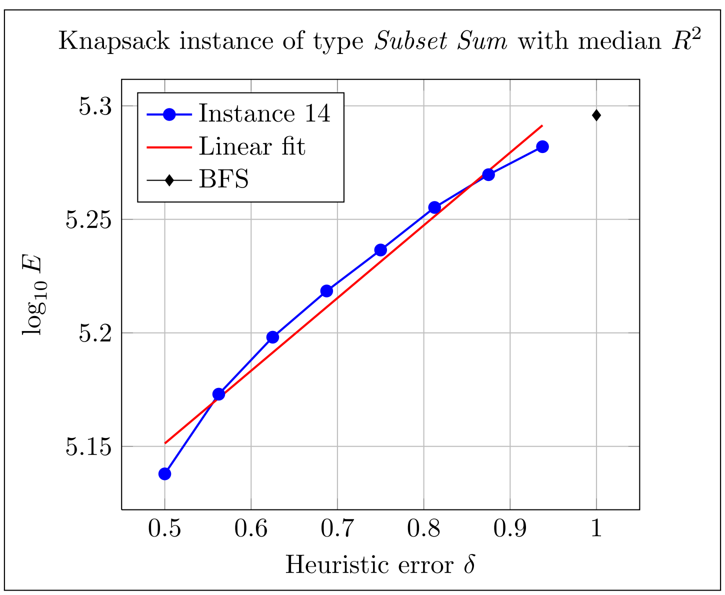

Figure 4: Graph of $\log_{10} E$ and its least-squares linear fit for the Knapsack instance of type Subset Sum with the median R^2 (see data in Table 5).

Remark Of course, there may be instances that poorly fit our prediction of linear dependence, such as instance #20 of Strongly Correlated type whose R^2 value is only 0.486, though those instances rarely show up in our experiments. In such an instance, the A^* search using heuristic function H_δ may explore even fewer nodes than the A^* search using $H_{\delta-\Delta}$ does, for some small $\Delta > 0$. This phenomenon can be explained by the degree to which we can control the accuracy of our heuristic function H_δ. In particular, we can only guarantee that H_δ is admissible and δ-approximate, while in reality it may provide an approximation better than δ to all nodes that are opened. Note that H_δ is not proportional to $(1-\delta)$. Hence, H_δ may be occasionally more accurate than $H_{\delta-\Delta}$ for some small $\Delta > 0$, resulting in fewer nodes expanded.

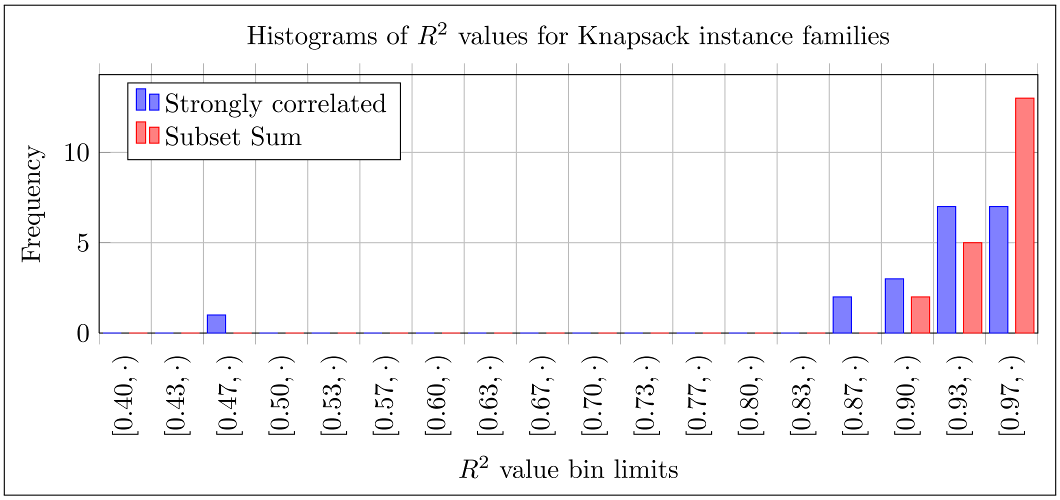

Figure 5: Histograms of the R^2 values for Knapsack instances.

To analyze more deeply how our data fit the model of Equation (17), we calculate the slope of the least-squares linear fit of $\log_{10} E$ for each Knapsack instance of type Strongly Correlated. Note that for such an instance, every search has the same optimal solution depth, denoted d, and thus, $\overline{d} = d$. Our data, given in Figure 6, show that for all but one instance with the worst R^2 value, the slope a of the linear fit of $\log_{10} E$ is fairly close to $d \log_{10} b_{\mathrm{BFS}}$, which is the slope of the hypothesized line given in Equation (17). Specifically, for any Knapsack instance of type Strongly Correlated, except instance #20,

$$0.73 d \log_{10} b_{\mathrm{BFS}} \le a \le 1.63 d \log_{10} b_{\mathrm{BFS}}\,.$$

7.2 A^* Search for Partial Latin Square Completion

The experimental results discussed above for the Knapsack problem support the hypothesis of linear scaling (cf., Equation (1) or (10)). However, several structural features of the search space and heuristic are unknown: for example, we cannot rule out the possibility that the approximation algorithm, when asked to produce an ϵ-approximation, does not in fact produce a significantly better approximation; likewise, we have no explicit control on the number of near-optimal solutions. In order to explore the hypothesis in more detail, we experimentally and analytically investigate a search space for the *partial Latin square completion problem* in which we can provide precise analytic control of heuristic error δ as well as the number of δ-optimal solutions N_δ.

7.2.1 The Partial Latin Square completion (PLS) Problem

A *Latin square* of order n is an $n \times n$ table in which each row and column is a permutation of the set $[n] = \{1, \ldots, n\}$. If only a few cells in an $n \times n$ table are filled with values from $[n]$ in such a

Knapsack instance type: Strongly Correlated

Instance	Optimal solution depth d	Effective branching factor of breath first search b_{BFS}	Slope of linear fit a	$a/(d \log_{10} b_{\mathrm{BFS}})$	Coefficient of determination R^2
1	11	4.2092	7.6583	1.1154	0.9395
2	9	5.3928	4.8966	0.7435	0.9183
3	6	10.8551	10.1038	1.6260	0.9647
4	7	8.2380	6.4279	1.0027	0.9710
5	7	8.0194	5.0882	0.8040	0.9161
6	6	10.6780	6.4511	1.0454	0.9696
7	7	8.7068	7.9087	1.2021	0.9436
8	8	6.7742	6.5616	0.9872	0.9782
9	6	11.4102	8.6847	1.3690	0.9571
10	9	5.5412	6.3690	0.9517	0.9461
11	7	8.3260	9.7685	1.5161	0.9689
12	5	18.0486	7.7848	1.2392	0.9314
13	7	8.0308	6.0376	0.9533	0.9646
14	5	15.0964	7.3004	1.2385	0.9676
15	6	10.0070	4.4219	0.7368	0.8788
16	9	5.7863	7.1815	1.0466	0.8698
17	7	8.3155	9.1738	1.4247	0.9498
18	8	6.9106	9.2837	1.3823	0.9729
19	7	8.3602	7.1807	1.1123	0.9770
20	7	7.0964	1.0055	0.1688	0.4860

Figure 6: Slopes of the least-squares linear fits of $\log_{10} E$ (varying δ) for the Knapsack instances of type Strongly Correlated. Details of these least-squares linear fits are given in Tables 1, 2, and 3. The R^2 values for these Knapsack instances are also included in this figure.

way that no value appears twice in a single row or column, then the table is called a *partial Latin square*. A *completion* of a partial Latin square L is a Latin square that can be obtained by filling the empty cells in L, see Figure 7 for an example. Note that not every partial Latin square has a completion. Since the problem of determining whether a partial Latin square has a completion is NP-complete (Colbourn, 1984), its search version (denoted PLS), i.e., given a partial Latin square L find a completion of L if one exists, is NP-hard.

1	**2**			
		5	**1**	**4**
3			**2**	
	1			**3**
	4			

1	**2**	3	4	5
2	3	**5**	**1**	**4**
3	5	4	**2**	1
4	**1**	2	5	**3**
5	**4**	1	3	2

Figure 7: A 5×5 partial Latin square (right) and its unique completion (left).

The PLS problem (also known as *partial quasi-group completion*) has been used in the recent past as a source of benchmarks for the evaluation of search techniques in constraint satisfaction and Boolean satisfiability (Gomes & Shmoys, 2002). Indeed, partially filled Latin squares carry embedded structures that are the trademark of real-life applications in scheduling and time-tabling. Furthermore, hard instances of the partially filled Latin square trigger "heavy-tail" behaviors in backtrack search algorithms which are common-place in real-life applications and require randomization and or restarting (Gomes, Selman, & Kautz, 1998). Additionally, the PLS problem exhibits a strong phase transition phenomena at the satisfiable/unsatisfiable boundary (when 42% of the cells are filled) which can be exploited to produce hard instances. We remark that the underlying structure of Latin squares can be found in other real-word applications including scheduling, timetabling (Tay, 1996), error-correcting code design, psychological experiments design and wavelength routing in fiber optics networks (Laywine & Mullen, 1998; Kumar, Russell, & Sundaram, 1996).

7.2.2 A Search Model for PLS

Fix a partial Latin square L of order n with $c > 0$ completions. We divide the cells of the $n \times n$ table into two types: the *black* cells, those that have been filled in L, and the *white* cells, those that are left blank in L. Let k be the number of white cells. The white cells are indexed from 0 to $k-1$ in a fixed order, e.g., left to right and top to bottom of the table. The task of A^* search now is to find a completion of L. Hard instances are obtained when the *white* cells are uniformly distributed within every row and every column and when the density of black cells is $(n^2 - k)/n^2 \approx 42\%$ to tap into the phase transition. We further insure that the number of completions is $c = O(1)$ (c is exactly 1 for the experiments).

To structure the search space for this problem, we place the white cells on a virtual circle so that the white cells of index i and $(i+1) \bmod k$ are adjacent. We can move along the circle, each step is either forward (from a white cell of index i to the cell of index $(i+1) \bmod k$) or backward (from a white cell of index i to the cell of index $(i-1) \bmod k$) and may set the content of the current cell. Formally, we define the search graph, denoted G_L, for the PLS instance given by L as follows: Each state (or node) of G_L is a pair (α, p), in which $p \in \{0, \ldots k-1\}$ indicates the index of the current white cell, and $\alpha : \{0, \ldots, k-1\} \to \{0, \ldots, n\}$ is a function representing the current assignment of values to the white cells (we adopt the convention that $\alpha(j) = 0$ means the white cell of index j has not been filled). There is a directed link (or edge) from state (α, p) to state (β, q) in the search graph G_L if and only if $q = (p \pm 1) \bmod k$, $\beta(q) \neq 0$, and $\alpha(j) = \beta(j)$ for all $j \neq q$. In other words, the link from state (α, p) to state (β, q) represents the step consisting of moving from the white cell of index p to the white cell of index q, and setting the value $\beta(q)$ to the white cell of index q. Figure 8 illustrates the links from one state to another in G_L. The cost of every link in G_L is a unit. Obviously, this search graph is regular and has (out-)degree of $2n$.

The starting state is $(\alpha_0, 0)$ where $\alpha_0(j) = 0$ for all j. A goal state (or solution) is of the form (α^*, p), where α^* is the assignment corresponding to a completion of L, and $p \in \{0, \ldots, k-1\}$. So, a solution on the cover tree of G_L is a path in the search graph G_L from the starting state to a goal state, and the length of an optimal solution is equal to k. We will show that the number of δ-optimal solutions in the cover tree of G_L is not too large.

Lemma 7.1. *Let L be an $n \times n$ partial Latin square with k white cells. Let α^* be the assignment corresponding to a completion of L. For any $0 \le t < k$, the number of paths of length $k+t$ in G_L from the starting state to a goal state of the form $(\alpha^*, \cdot)$ is no more than $2\left(t + 2 + t\binom{k+t}{t}\right) n^t$.*

Proof. We represent a path in G_L of length $k+t$ from the starting state as a pair $\langle P, \vec{v} \rangle$, in which P is a $(k+t)$-length path in the circle of white cells starting from the white cell of index 0, and $\vec{v} = (v_1, \ldots, v_{k+t})$ is a sequence of values in $[n]$ with v_i being the value assigned to the white cell visited at the i^{th} step of the path P. Consider a pair $\langle P, \vec{v} \rangle$ that represents a path in G_L ending up at a goal state $(\alpha^*, \cdot)$. Since $\alpha^*(j) \neq 0$ for all j, every white cell must be visited at some non-zero step of P. Let $s_j > 0$ be the last step at which the white cell of index j is visited. Then we must have $v_{s_j} = \alpha^*(j)$ for all $j \in \{0, \ldots, k-1\}$. Given such a path P, there are n^t ways of assigning values to the white cells in order to eventually obtain the assignment α^*. Thus, the number of $(k+t)$-length paths in G_L from the starting state to a goal state $(\alpha^*, \cdot)$ is equal to $|\mathcal{P}_t| n^t$, where $\mathcal{P}_t$ is the set of $(k+t)$-length paths on the circle of white cells that start at white cell of index 0 and visit every white cell.

It remains to upper bound $|\mathcal{P}_t|$. Consider a path $P \in \mathcal{P}_t$; our strategy is to bound the number of backward (or forward) steps in P. As $t < k$, there are at least $k - t \geq 1$ white cells visited exactly

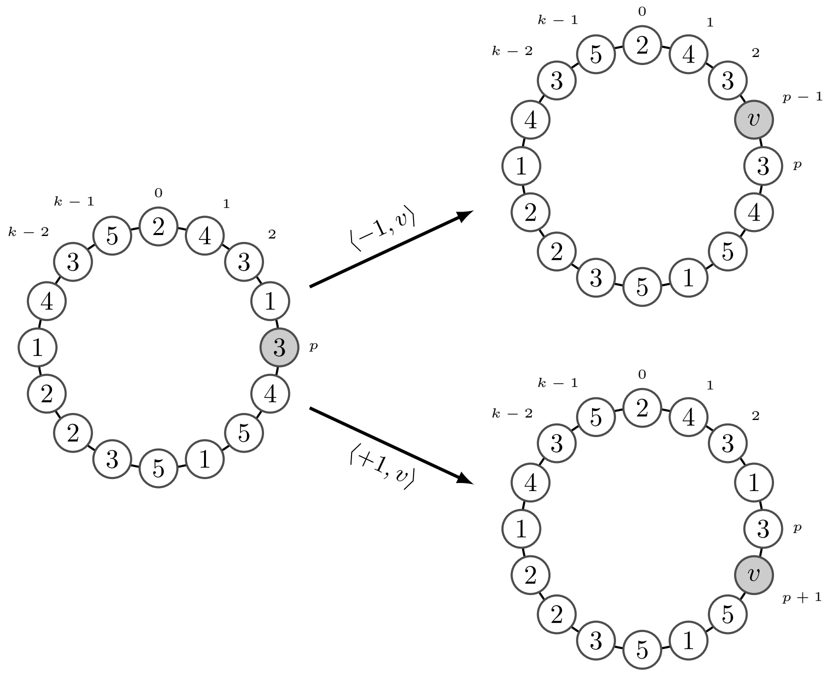

Figure 8: The links connecting states in a PLS search graph. The label $\langle +1, v\rangle$ (resp., $\langle -1, v\rangle$) on the links means moving forward (resp., backward) and setting value $v \in [n]$ to the next white cell.

once in P. Let w be the index of a white cell that is visited exactly once in P and let s be the step at which the white cell w is visited.

Assume the step s is a forward step, i.e., the white cell visited at step $s-1$ is $(w-1) \bmod k$. Let w_0 be the farthest white cell from w in the backward direction that is visited before step s. Precisely, $w_0 = (w-\ell) \bmod k$, where ℓ is the maximal number in $\{0, \dots, k-1\}$ for which the white cell $(w-\ell) \bmod k$ is visited before step s. Let $w_j = (w_0 + j) \bmod k$, for $j = 0, \dots, k-1$. Note that $w_\ell = w$. Then the set of white cells visited at the first s steps is $\{w_0, w_1, \dots, w_\ell\}$, and by deleting some steps among the first s steps in P we will obtain the path $(w_0, w_1, \dots, w_\ell)$ from w_0 to w_ℓ in the forward direction. Each of the white cells $w_{\ell+1}, \dots, w_{k-1}$ must be visited at a step after step s and also in the forward direction because the white cell w_ℓ is visited only once and at a forward step. Thus, by deleting t steps from P we obtain a path visiting the white cells $w_0, w_1, \dots, w_{k-1}$ in the forward direction. Let $s_0, \dots, s_{k-1}$ be the steps in P that are not deleted, where w_j is visited at step s_j in P, and $1 \le s_0 < s_1 < \ldots < s_{k-1} \le k+t$. Then steps $s_1, \dots, s_{k-1}$ are all forward steps (step s_0 can be forward or backward). Moreover, the number of backward steps and the number of forward steps between s_{j-1} and s_j must be equal for all $j = 1, \dots, k-1$. Let Δ be the number of deleted steps before s_0 and after s_{k-1} so that there are exactly $(t-\Delta)/2$ backward steps between s_0 and s_k. This shows there are at most $\Delta + 1 + (t-\Delta)/2 = 1 + (t+\Delta)/2 \le t+1$ backward steps in P. Note that there at most $\binom{k+t}{j}$ paths in $\mathcal{P}_t$ that have exactly j backward steps. Path P has $t+1$ backward steps only when $\Delta = t$ (and thus $s_j = s_{j-1}+1$ for all $j = 1, \dots, k-1$) and every step from 1 to s_0 and after s_{k-1} is backward. There are $t+1$ such paths in $\mathcal{P}_t$, each corresponding to a choice of $s_0 \in \{1, \dots, t+1\}$.

Similarly, if the step s is a backward step, then there are at most $t+1$ forward steps in P. Also, there are $t+1$ paths in $\mathcal{P}_t$ that have exactly $t+1$ forward steps, and at most $\binom{k+t}{j}$ paths in $\mathcal{P}_t$ that have exactly j forward steps. Hence,

$$|\mathcal{P}_t| \le 2\left(t+1+\sum_{j=0}^{t}\binom{k+t}{j}\right) \le 2\left(t+2+t\binom{k+t}{t}\right).$$

The last inequality holds since the coefficient $\binom{k+t}{j}$ increases as j increases for $j < (k+t)/2$. □

The upper bound in Lemma 7.1 is achieved when $t = 0$. In fact, there are four ways to visit every white cell in k steps starting from the white cell 0: taking either k forward steps or k backward steps or one backward step followed by $k-1$ forward steps or one forward steps followed by $k-1$ backward steps. So the number of optimal solutions in the cover tree of G_L is equal to $4c$, since there are c completions of the initial partial Latin square.

Theorem 7.1. *Let L be an $n \times n$ partial Latin square with k white cells and c completions. For any $0 < \delta < 1$, the number of nodes expanded by A^* search on G_L with a δ-accurate heuristic is no more than $B(\delta)$, where*

$$B(\delta) = \begin{cases} 2(2n)^{\delta k} + 4ck & \text{if } \delta k < 1\,, \\ 2(2n)^{\delta k} + 4ck\left(\lfloor \delta k\rfloor + 2 + \lfloor \delta k\rfloor \binom{k+\lfloor \delta k\rfloor}{\lfloor \delta k\rfloor}\right) n^{\lfloor \delta k\rfloor} & \text{if } \delta k \ge 1\,. \end{cases}$$

Proof. By Lemma 3.1, the number of nodes expanded by A^* search on G_L with a δ-accurate heuristic is upper-bounded by $2(2n)^{\delta k} + kN_\delta$, where N_δ is the number of δ-optimal solutions in the cover tree of G_L. So, we only need to bound N_δ.

If $\delta k < 1$, then N_δ equals the number of optimal solutions, which implies the upper bound of $2(2n)^{\delta k} + 4ck$ on the number of expanded nodes by A^*.

In the general case, let $\ell = \lfloor \delta k \rfloor$. Since $\delta k < k$, by Lemma 7.1, we have

$$\begin{aligned} N_\delta &\le c\sum_{t=0}^{\ell} 2\left(t+2+t\binom{k+t}{t}\right)n^t \\ &\le 2c\sum_{t=0}^{\ell}(t+2)n^t + 2c\binom{k+\ell}{\ell}\left(\sum_{t=0}^{\ell} tn^t\right) \\ &\le 4c(\ell+2)n^\ell + 4c\binom{k+\ell}{\ell}\ell n^\ell\,. \end{aligned}$$

The second inequality holds because $\binom{k+t}{t} \le \binom{k+\ell}{\ell}$ for all $t \le \ell$. The last inequality is obtained by applying the fact that $\sum_{t=0}^{\ell} tn^t \le 2\ell n^\ell$ and $\sum_{t=0}^{\ell} n^t \le 2n^\ell$ for all integers $n \ge 2$ and $\ell \ge 0$, which can be proved easily by induction on ℓ. Hence, the number of nodes expanded by A^* is no more than

$$2(2n)^{\delta k} + 4ck\left(\ell + 2 + \ell\binom{k+\ell}{\ell}\right)n^\ell\,.$$

□

Corollary 7.1. *Suppose $0 < \delta < 1$. Then the number of nodes expanded by A^* search on G_L with a δ-accurate heuristic is*

$$O\left(k^{3/2}\,(1+\delta)^k\,(1+{}^{1}\!/\!{}_{\delta})^{\delta k}\,n^{\delta k}\right).$$

Proof. By Theorem 7.1, all we need is an upper bound on the binomial coefficient $\binom{k+\ell}{\ell}$ for large k, where $\ell = \lfloor \delta k \rfloor$. Since both k and ℓ are large, we will bound this binomial coefficient using Stirling's formula, which asserts that $n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$. More precisely, write $n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \lambda_n$, then $\lambda_n \to 1$ as $n \to \infty$. We have

$$\begin{aligned}
\binom{k+\ell}{\ell} &= \frac{(k+\ell)!}{k!\ell!} \\
&= \frac{\sqrt{2\pi(k+\ell)}\left(\frac{k+\ell}{e}\right)^{k+\ell}\lambda_{k+\ell}}{\sqrt{2\pi k}\left(\frac{k}{e}\right)^k \lambda_k \cdot \sqrt{2\pi\ell}\left(\frac{\ell}{e}\right)^\ell \lambda_\ell} \\
&= \frac{\lambda_{k+\ell}}{\lambda_k\lambda_\ell} \cdot \frac{\sqrt{k+\ell}}{\sqrt{2\pi k\ell}} \cdot \frac{(k+\ell)^{k+\ell}}{k^k\ell^\ell}.
\end{aligned}$$

By Stirling's formula, the term $\lambda_{k+\ell}/\lambda_k\lambda_\ell$ is $O(1)$. Since

$$\frac{k+\ell}{\ell} = 1 + \frac{k}{\lfloor \delta k \rfloor} \le 1 + \frac{k}{\delta k - 1} \le 1 + 2/\delta$$

for $k > 2/\delta$, the term $\sqrt{k+\ell}/\sqrt{2\pi k\ell}$ is $O(1/\sqrt{k})$. The remaining term is

$$\frac{(k+\ell)^{k+\ell}}{k^k\ell^\ell} = \left(1 + \frac{\ell}{k}\right)^k \left(1 + \frac{k}{\ell}\right)^\ell \le (1+\delta)^k \left(1 + \frac{1}{\delta}\right)^{\delta k}$$

since $\ell \le \delta k$ and the function $g(x) = (1 + k/x)^x$ monotonically increases for $x > 0$. Hence,

$$\binom{k+\ell}{\ell} = O\left(\frac{1}{\sqrt{k}}(1+\delta)^k \left(1 + \frac{1}{\delta}\right)^{\delta k}\right).$$

From Theorem 7.1, the number of nodes expanded by A^* is no more than

$$\begin{aligned}
B(\delta) &= 2(2n)^{\delta k} + O\left(k^2 \binom{k+\ell}{\ell} n^{\delta k}\right) \\
&= O\left(k^{3/2}(1+\delta)^k \left(1 + \frac{1}{\delta}\right)^{\delta k} n^{\delta k}\right).
\end{aligned}$$

□

It follows from the above corollary that the effective branching factor of A^* using a δ-accurate heuristic on G_L is asymptotically at most $(1+\delta)\left(1 + {}^1\!/\!_\delta\right)^\delta n^\delta$, which is significantly smaller than the brute-force branching factor of $2n$, since both $(1+\delta)n^\delta$ and $\left(1 + {}^1\!/\!_\delta\right)^\delta$ converge to 1 as $\delta \to 0$.

7.2.3 Experiments

Given the search model for the PLS problem described above, we provide experimental results of A^* search on a few PLS instances, each of which is determined by a large partial Latin square with a single completion. For each PLS instance in our experiments, we run A^* search with different heuristics of the form $(1-\delta)h^*$ given by various values of $\delta \in [0, 1)$. We emphasize that by the *dominance* property of admissible heuristics, the number of nodes expanded by A^* using any admissible δ-accurate heuristic strictly larger than $(1-\delta)h^*$ is less than or equal to that by the A^* using the heuristic $(1-\delta)h^*$. In other words, the heuristic $(1-\delta)h^*$ is *worse* than most admissible δ-accurate heuristics.

To build the oracle for the heuristic $(1-\delta)h^*$ on a search graph G_L, we use the information about the completion of the partial Latin square L to compute h^*. Consider a partial Latin square L with k white cells, and an arbitrary state (α, p) in G_L. We will show how to compute the optimal cost $h^*(\alpha, p)$ to reach a goal state in G_L from state (α, p). Let $X(\alpha)$ be the set of white cells at which α disagrees with the completion of L, then $h^*(\alpha, p)$ is equal to the length of the shortest paths on the cycle starting from p and then visiting every point in $X(\alpha)$. The case in which $|X(\alpha) \setminus \{p\}| \leq 1$ is easy to handle, so we shall assume $|X(\alpha) \setminus \{p\}| \geq 2$ from now on. In particular, suppose $X(\alpha) \setminus \{p\} = \{p_1, \ldots, p_\ell\}$ with $\ell > 1$, where p_j is the j^{th} point in $X(\alpha) \setminus \{p\}$ that is visited when moving forward (clockwise) from p; see Figure 9. There are two types of paths on the cycle starting from p and visiting every point in $X(\alpha) \setminus \{p\}$: type **I** includes those that do not visit p, type **II** includes those visiting p. Let ℓ_1 and ℓ_2 be the length of the shortest paths of type **I** and type **II**, respectively. Then

$$h^*(\alpha, p) = \begin{cases} \min\{\ell_1, \ell_2\} & \text{if } p \notin X(\alpha)\,, \\ \min\{\ell_1 + 2, \ell_2\} & \text{if } p \in X(\alpha)\,. \end{cases}$$

So now we only need to compute ℓ_1 and ℓ_2. Computing ℓ_1 is straightforward: it is realized by either moving forward from p to p_ℓ or moving backward from p to p_1. That is

$$\ell_1 = \min\{\overline{p_\ell - p}, \overline{p - p_1}\}$$

where $\overline{z} \overset{\text{def}}{=} z \bmod k$ for any integer z. To compute ℓ_2, we consider two options for each j: option *(a)* moving forward from p to p_j and then moving backward from p_j to p_{j+1}, option *(b)* moving backward from p to p_{j+1} and then moving forward from p_{j+1} to p_j. Thus,

$$\ell_2 = \min_{1 \leq j < \ell} \Big(\min\{\overline{p_j - p}, \overline{p - p_{j+1}}\} + \overline{p_j - p_{j+1}} \Big)\,.$$

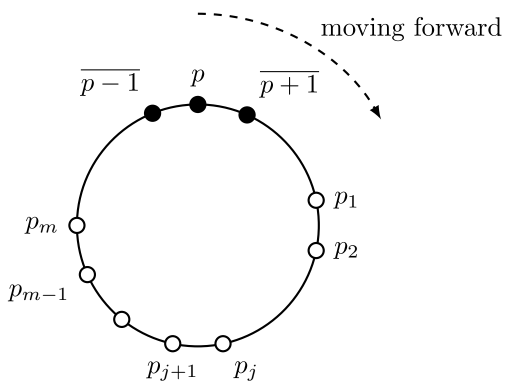

Figure 9: Layout of the points in $X(\alpha)$.

Now we describe our experiments in detail. We generate six partial Latin squares with orders from 10 to 20 in the following way. Initially, we generate several partial Latin squares obtained at or near the phase transition with white cells uniformly distributed within every row and column. Each instance is generated from a complete Latin square with a suitably chosen random subsets of its cells cleared. Each candidate partial Latin square is solved again with an exhaustive backtracking search method to find all completions. The subset of candidates with exactly one completion is retained for the experiments. For each partial Latin square L and each chosen value of δ, we record the total number E of nodes expanded by A^* on the search graph G_L with the $(1-\delta)h^*$ heuristic. Then, as in the Knapsack experiments, the effective branching factor of A^* is calculated as $E^{1/k}$, since the optimal solution depth in G_L equals the number of white cells in L. Our first purpose is to compare these effective branching factors obtained from experiments to our upper bound obtained from theoretical analysis. Recall from Theorem 7.1 that $E \le B(\delta)$, where in this case

$$B(\delta) = \begin{cases} 2(2n)^{\delta k} + 4k & \text{if } \delta k < 1\,, \\ 2(2n)^{\delta k} + 4k\left(\lfloor \delta k \rfloor + 2 + \lfloor \delta k \rfloor \binom{k+\lfloor \delta k \rfloor}{\lfloor \delta k \rfloor}\right) n^{\lfloor \delta k \rfloor} & \text{if } \delta k \ge 1\,. \end{cases}$$

Therefore, we calculate the theoretical upper bound $B(\delta)^{1/k}$ on the effective branching factor $E^{1/k}$. For deeper comparison, we calculate the multiplicative gap $B(\delta)^{1/k}/E^{1/k}$ between our theoretical bound and the actual values. In our empirical results given in Tables 7 and 8, these multiplicative gaps are close to 1 when δ is small and k is large. Notice that for each given k, the upper bounds of $B(\delta)$ are almost the same for the δ's with the same value of $\lfloor \delta k \rfloor$. This is why the multiplicative gaps for those δ's sometimes increase when δ decrease. However, the multiplicative gaps decrease as $\lfloor \delta k \rfloor$ decreases, for each fixed k. Our upper bounds in the cases with $\delta k < 1$ are much tighter than in the others (with the same k) because in the cases of $\delta k < 1$ we can compute the number of δ-optimal solutions exactly. Also observe that, for each fixed δ, the multiplicative gaps decrease as k increases. Finally, the experiments show a dramatic gap between the effective branching factors and the corresponding brute-force branching factor, which equals $2n$. In fact, for each instance, both the effective branching factor $E^{1/k}$ and our theoretical upper bound $B(\delta)^{1/k}$ approach 1 as δ approaches 0.

As in the experiments for the Knapsack problem, our data for the partial Latin square problem also support the linear dependance of $\log E$ on δ. In particular, all but one partial Latin square instances have the R^2 larger than 0.9 (the worst one has R^2 value equal to 0.8698). The median R^2 value for our partial Latin square instances is 0.9304. The graph for the instance with the median R^2 is shown in Figure 10.

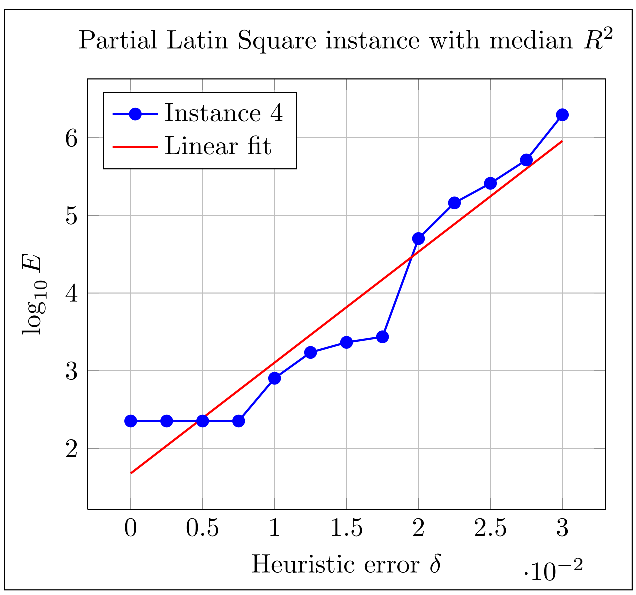

Figure 10: Graph of $\log_{10} E$ and its least-squares least-squares linear fit (or "Linear fit") for the partial Latin square instance with the median R^2 (see data in Table 8).

We also investigate how the slope of the least-squares linear fit of $\log E$ approximates the slope of $d \log b$ in the hypothesized linear dependence of Equation (10). Recall that in this case, the branching factor is $b = 2n$ and the optimal solution depth is $d = k$. Figure 11 shows that, for every PLS instance in our experiment, the slope α of the least-squares linear fit of $\log_{10} E$ approximates to $k \log_{10}(2n)$ by a factor of 0.8, i.e., $\alpha \approx 0.8k \log_{10}(2n)$. In other words, our experimental results for the PLS indicate the following relationship:

$$\log_{10} E \approx \delta \cdot 0.8k \log_{10}(2n) + \xi\,, \quad \text{or equivalently,} \quad E \approx (2n)^{0.8\delta k}\,.$$

Thus, empirically, the effective branching factor of A^* search using the heuristic $(1-\delta)h^*$ on the given PLS search space approximates to $(2n)^{0.8\delta}$. By the dominance property of admissible heuristics, this is also an empirical upper bound on the effective branching factor of A^* using any admissible δ-accurate on the same search space.

Instance #	n	k	Slope of linear fit line α	$\alpha/(k \log_{10}(2n))$
1	10	44	43.3901	0.7580
2	12	63	73.7527	0.8482
3	14	86	98.3613	0.7903
4	16	113	142.7056	0.8390
5	18	143	179.1665	0.8050
6	20	176	225.4152	0.7995

Figure 11: Slopes of the least-squares linear fits of $\log_{10} E$ for the partial Latin square instances.

8. Reduction in Depth vs. Branching Factor; Comparison with Previous Work

In this section we compare our results with those obtained by Korf et al. (Korf & Reid, 1998; Korf et al., 2001). As mentioned in the introduction, they concluded that "the effect of a heuristic function is to reduce the effective depth of a search rather than the effective branching factor." Considering the striking qualitative difference between their findings and ours, it seems interesting to discuss why their conclusions do not apply to accurate heuristics.

They study the b-ary tree search model, as above, and permit multiple solutions. However, their analysis depends critically on the following equilibrium assumption:

Equilibrium Assumption: The number of nodes at depth i with heuristic value not exceeding ℓ is $b^i P(\ell)$, where $P(\ell)$ is the probability that $h(v) \leq \ell$ when v is chosen uniformly at random among all nodes of given depth, *in the limit of large depth.*

We remark that while the equilibrium assumption is a strong structural requirement, it holds *in expectation* for a rich class of "symmetric" search spaces. To be specific, for any state-transitive search space,[4] like the Rubik's cube, the quantity $b^i P(\ell)$ is precisely the *expected* number of vertices at depth i with $h(v) \leq \ell$ if the goal state (or initial state) is chosen uniformly at random. Korf et al. (2001) observe that under the equilibrium assumption, one can directly control the number of expanded nodes of total weight no more than ℓ, a quantity we denote $E(\ell)$: indeed, in this case $E(\ell) = \sum_{i \leq \ell} b^i P(\ell - i)$. With this in hand, they consider the ratio

$$\frac{E(\ell)}{E(\ell-1)} = \frac{\sum_{i=0}^{\ell} b^i P(\ell - i)}{\sum_{i=0}^{\ell-1} b^i P(\ell - 1 - i)} = b \cdot \frac{\sum_{i=0}^{\ell} b^{i-1} P(\ell - i)}{\sum_{i=1}^{\ell} b^{i-1} P(\ell - i)} \geq b\,, \tag{18}$$

and conclude that $E(d) \geq b^{d-1} E(1)$; thus the effective branching factor is

$$\sqrt[d]{b^{d-1} E(1)} \approx b \sqrt[d]{E(1)}$$

4. We say that a search space is state-transitive if the structure of the search graph is independent of the starting node. Note that any Cayley graph has this property, so natural search spaces formed from algebraic problems like the Rubik's cube or 15-puzzle, with the right choice of generators, have this property.

if the optimal solution lies at depth d.

A difficulty with this approach is that even in the presence of a mildly accurate heuristic satisfying, for example,

$$h(v) \geq \epsilon h^*(v) \quad \text{for small, constant, } \epsilon > 0\,,$$

the actual values of these quantities satisfy

$$E(1) = E(2) = \cdots = E(t) = 0$$

for all $t \leq \epsilon d$. (Even the root of the tree has $h(\text{root}) \geq \epsilon \cdot d$.) Observe, then, that if $E(\epsilon d) = 1$ the argument above actually results in an effective branching factor of $\sqrt[d]{b^{d-\epsilon d} E(\epsilon d)} = \sqrt[d]{b^{(1-\epsilon)d}} = b^{1-\epsilon}$, yielding reduction in the branching factor. Indeed, applying this technique to infer estimates on the complexity of A^*, even assuming the equilibrium assumption, appears to require control of the threshold quantity ℓ_0 at which the quantities $\sum b^i P(\ell_0 - i)$ become non-negligible. Of course, the equilibrium assumption may well apply to heuristics with weaker or, for example, nonuniform accuracy.

One perspective on this issue can be obtained by considering the case of search on a b-regular (non-bipartite, connected) graph $G = (V, E)$ and observing that the selection of a node "uniformly at random from all nodes of a given depth, in the limit of large depth" is, in this case, equivalent to selection of a random node in the graph. If we again consider a mildly accurate heuristic h for which, say, $h(v) \geq \epsilon h^*(v)$ for a small constant ϵ, we have $b^i P(\ell) \leq b^i \Pr_v[\epsilon \cdot \text{dist}(v, S) \leq \ell]$, where v is chosen uniformly at random in the graph, S is the set of solution nodes, and $\text{dist}(v, S)$ denotes the length of the shortest path from v to a node of S. As

$$\Pr_v[\text{dist}(v, S) \leq \ell/\epsilon] = \frac{|\{v \mid \text{dist}(v, S) \leq \ell/\epsilon\}|}{|V|} \leq \frac{|S| \cdot b^{\ell \cdot \epsilon^{-1}}}{|V|}$$

in any b-regular graph, we can only expect the relation of equation (18) to hold past the threshold value $\ell_0 \approx \epsilon \log_b(|S|/|V|)$.

Acknowledgments

We wish to thank anonymous reviewers for their constructive comments. Author Hang Dinh was supported by IU South Bend Faculty Research Grant. Author Laurent Michel was supported by the NSF under grant #0642906. Author Alexander Russell was supported by the NSF under grant #1117426.

Appendix A: Tables of Experimental Results

#	n	Heuristic error δ	Total node expansions E	Optimal solution depth d	Search time (seconds)	$h^*([n])/m$	Effective branching factor $\sqrt[d]{E}$	$\log_{10} E$	Linear fit to $\log_{10} E$	R^2
1	23	0.5	5627	11	125	10887/200	2.192473	3.7503	3.7956	
	23	0.5625	5882	11	101	10887/200	2.201325	3.7695	4.2742	
	23	0.625	167660	11	858	10887/200	2.985026	5.2244	4.7529	
	23	0.6875	211946	11	744	10887/200	3.049315	5.3262	5.2315	
	23	0.75	772257	11	1341	10887/200	3.42966	5.8878	5.7102	
	23	0.8125	1470135	11	1318	10887/200	3.636376	6.1674	6.1888	
	23	0.875	6118255	11	2025	10887/200	4.139674	6.7866	6.6674	
	23	0.9375	7154310	11	1653	10887/200	4.198968	6.8546	7.1461	
	23	BFS	7347748	11	1101		4.209164	6.8662		0.9395
2	23	0.5	44481	9	622	7820/157	3.284459	4.6482	4.7018	
	23	0.5625	45537	9	507	7820/157	3.293033	4.6584	5.0078	
	23	0.625	372163	9	1497	7820/157	4.158822	5.5707	5.3139	
	23	0.6875	474221	9	1293	7820/157	4.272327	5.6760	5.6199	
	23	0.75	1358735	9	1751	7820/157	4.802412	6.1331	5.9259	
	23	0.8125	2508134	9	1734	7820/157	5.140898	6.3994	6.2320	
	23	0.875	3508255	9	1469	7820/157	5.336203	6.5451	6.5380	
	23	0.9375	3569052	9	1047	7820/157	5.3464	6.5526	6.8441	
	23	BFS	3857597	9	566		5.392783	6.5863		0.9183
3	23	0.5	94	6	6	5991/121	2.132331	1.9731	1.9674	
	23	0.5625	125	6	7	5991/121	2.236068	2.0969	2.5989	
	23	0.625	5528	6	98	5991/121	4.204955	3.7426	3.2304	
	23	0.6875	9002	6	105	5991/121	4.560962	3.9543	3.8619	
	23	0.75	31800	6	206	5991/121	5.628654	4.5024	4.4934	
	23	0.8125	109080	6	301	5991/121	6.912326	5.0377	5.1248	
	23	0.875	879884	6	707	5991/121	9.788983	5.9444	5.7563	
	23	0.9375	1477032	6	560	5991/121	10.671652	6.1694	6.3878	
	23	BFS	1636093	6	224		10.855121	6.2138		0.9647
4	23	0.5	3696	7	86	6343/154	3.233523	3.5677	3.7471	
	23	0.5625	21847	7	256	6343/154	4.16786	4.3394	4.1489	
	23	0.625	44166	7	303	6343/154	4.608759	4.6451	4.5506	
	23	0.6875	53464	7	258	6343/154	4.73628	4.7281	4.9524	
	23	0.75	253321	7	553	6343/154	5.914977	5.4037	5.3541	
	23	0.8125	760792	7	788	6343/154	6.921191	5.8813	5.7558	
	23	0.875	1975195	7	957	6343/154	7.93182	6.2956	6.1576	
	23	0.9375	2317663	7	694	6343/154	8.115082	6.3651	6.5593	
	23	BFS	2574876	7	383		8.23801	6.4108		0.9710
5	23	0.5	23645	7	305	6785/205	4.215217	4.3737	4.3803	
	23	0.5625	30501	7	285	6785/205	4.371357	4.4843	4.6983	
	23	0.625	72597	7	429	6785/205	4.947855	4.8609	5.0163	
	23	0.6875	308417	7	754	6785/205	6.083628	5.4891	5.3343	
	23	0.75	968504	7	1074	6785/205	7.164029	5.9861	5.6523	
	23	0.8125	1681026	7	1047	6785/205	7.751179	6.2256	5.9703	
	23	0.875	1833872	7	823	6785/205	7.848145	6.2634	6.2883	
	23	0.9375	1833644	7	585	6785/205	7.848005	6.2633	6.6064	
	23	BFS	2132977	7	306		8.019382	6.3290		0.9161
6	23	0.5	1981	6	46	5012/148	3.543894	3.2969	3.4645	
	23	0.5625	12316	6	139	5012/148	4.80557	4.0905	3.8677	
	23	0.625	21699	6	151	5012/148	5.281289	4.3364	4.2709	
	23	0.6875	26575	6	131	5012/148	5.462761	4.4245	4.6741	
	23	0.75	131561	6	290	5012/148	7.131615	5.1191	5.0773	
	23	0.8125	395118	6	431	5012/148	8.566192	5.5967	5.4805	
	23	0.875	1080314	6	547	5012/148	10.129585	6.0336	5.8837	
	23	0.9375	1282206	6	409	5012/148	10.423006	6.1080	6.2869	
	23	BFS	1482293	6	219		10.677978	6.1709		0.9696
7	23	0.5	1834	7	51	6187/122	2.925499	3.2634	2.8110	
	23	0.5625	1956	7	43	6187/122	2.952538	3.2914	3.3053	
	23	0.625	2039	7	36	6187/122	2.970119	3.3094	3.7996	
	23	0.6875	23275	7	159	6187/122	4.20573	4.3669	4.2939	
	23	0.75	30974	7	138	6187/122	4.380978	4.4910	4.7882	
	23	0.8125	173886	7	332	6187/122	5.605434	5.2403	5.2825	
	23	0.875	675468	7	526	6187/122	6.80457	5.8296	5.7768	
	23	0.9375	3440759	7	984	6187/122	8.586333	6.5367	6.2711	
	23	BFS	3793204	7	568		8.706789	6.5790		0.9436

Table 1: Results for the Knapsack instances of type Strongly Correlated.

#	n	Heuristic error δ	Total node expansions E	Optimal solution depth d	Search time (seconds)	$h^*([n])/m$	Effective branching factor $\sqrt[d]{E}$	$\log_{10} E$	Linear fit to $\log_{10} E$	R^2
8	23	0.5	8299	8	129	6400/153	3.089429	3.9190	3.7753	
	23	0.5625	8741	8	105	6400/153	3.109533	3.9416	4.1854	
	23	0.625	58455	8	335	6400/153	3.943235	4.7668	4.5955	
	23	0.6875	93500	8	332	6400/153	4.181686	4.9708	5.0056	
	23	0.75	216413	8	479	6400/153	4.644195	5.3353	5.4157	
	23	0.8125	536713	8	558	6400/153	5.202568	5.7297	5.8258	
	23	0.875	2569955	8	1066	6400/153	6.327624	6.4099	6.2359	
	23	0.9375	4096150	8	1027	6400/153	6.707288	6.6124	6.6460	
	23	BFS	4434697	8	655		6.7742	6.6469		0.9782
9	23	0.5	430	6	19	5835/121	2.747334	2.6335	2.3749	
	23	0.5625	460	6	16	5835/121	2.778388	2.6628	2.9177	
	23	0.625	5313	6	84	5835/121	4.177245	3.7253	3.4605	
	23	0.6875	9507	6	91	5835/121	4.602643	3.9780	4.0033	
	23	0.75	11268	6	75	5835/121	4.734867	4.0518	4.5461	
	23	0.8125	88158	6	229	5835/121	6.671297	4.9453	5.0889	
	23	0.875	790402	6	646	5835/121	9.615562	5.8978	5.6317	
	23	0.9375	2008558	6	673	5835/121	11.232611	6.3029	6.1745	
	23	BFS	2206805	6	334		11.410219	6.3438		0.9571
10	23	0.5	14162	9	192	6762/171	2.892252	4.1511	4.1618	
	23	0.5625	15321	9	162	6762/171	2.917641	4.1853	4.5599	
	23	0.625	178178	9	669	6762/171	3.832024	5.2509	4.9579	
	23	0.6875	214332	9	574	6762/171	3.911497	5.3311	5.3560	
	23	0.75	872080	9	1052	6762/171	4.571533	5.9406	5.7541	
	23	0.8125	2128661	9	1306	6762/171	5.048042	6.3281	6.1521	
	23	0.875	3942938	9	1379	6762/171	5.405911	6.5958	6.5502	
	23	0.9375	4543001	9	1118	6762/171	5.491674	6.6573	6.9482	
	23	BFS	4924992	9	721		5.541159	6.6924		0.9461
11	23	0.5	315	7	19	6465/106	2.274582	2.4983	2.1619	
	23	0.5625	330	7	15	6465/106	2.289748	2.5185	2.7724	
	23	0.625	974	7	29	6465/106	2.672619	2.9886	3.3830	
	23	0.6875	22374	7	232	6465/106	4.182076	4.3497	3.9935	
	23	0.75	26883	7	195	6465/106	4.293214	4.4295	4.6040	
	23	0.8125	199464	7	514	6465/106	5.716412	5.2999	5.2146	
	23	0.875	783863	7	751	6465/106	6.950792	5.8942	5.8251	
	23	0.9375	2579423	7	880	6465/106	8.240087	6.4115	6.4356	
	23	BFS	2773773	7	406		8.326044	6.4431		0.9689
12	23	0.5	1029	5	35	5073/106	4.003899	3.0124	2.5015	
	23	0.5625	1163	5	29	5073/106	4.103136	3.0656	2.9880	
	23	0.625	1310	5	25	5073/106	4.201983	3.1173	3.4746	
	23	0.6875	3968	5	50	5073/106	5.244624	3.5986	3.9611	
	23	0.75	14820	5	92	5073/106	6.826053	4.1708	4.4477	
	23	0.8125	75333	5	212	5073/106	9.449244	4.8770	4.9342	
	23	0.875	363263	5	380	5073/106	12.943277	5.5602	5.4208	
	23	0.9375	1710935	5	589	5073/106	17.646017	6.2332	5.9073	
	23	BFS	1915195	5	283		18.048562	6.2822		0.9314
13	23	0.5	6701	7	154	6072/122	3.520395	3.8261	3.6957	
	23	0.5625	7084	7	127	6072/122	3.548459	3.8503	4.0730	
	23	0.625	43514	7	379	6072/122	4.598978	4.6386	4.4504	
	23	0.6875	71911	7	383	6072/122	4.941148	4.8568	4.8277	
	23	0.75	85427	7	313	6072/122	5.064232	4.9316	5.2050	
	23	0.8125	376321	7	573	6072/122	6.259049	5.5756	5.5824	
	23	0.875	1441947	7	862	6072/122	7.583154	6.1589	5.9597	
	23	0.9375	1963475	7	655	6072/122	7.925079	6.2930	6.3371	
	23	BFS	2154280	7	324		8.030775	6.3333		0.9646
14	23	0.5	418	5	15	4636/140	3.343761	2.6212	2.8386	
	23	0.5625	3629	5	63	4636/140	5.151781	3.5598	3.2949	
	23	0.625	7016	5	74	4636/140	5.877842	3.8461	3.7512	
	23	0.6875	8503	5	62	4636/140	6.108217	3.9296	4.2075	
	23	0.75	51480	5	162	4636/140	8.756443	4.7116	4.6637	
	23	0.8125	178163	5	258	4636/140	11.22441	5.2508	5.1200	
	23	0.875	550403	5	352	4636/140	14.064884	5.7407	5.5763	
	23	0.9375	668276	5	246	4636/140	14.621475	5.8250	6.0326	
	23	BFS	784088	5	110		15.096385	5.8944		0.9676

Table 2: Results for the Knapsack instances of type Strongly Correlated.

#	n	Heuristic error δ	Total node expansions E	Optimal solution depth d	Search time (seconds)	$h^*([n])/m$	Effective branching factor $\sqrt[d]{E}$	$\log_{10} E$	Linear fit to $\log_{10} E$	R^2
15	23	0.5	15713	6	218	5825/211	5.004682	4.1963	4.3104	
	23	0.5625	17658	6	184	5825/211	5.102977	4.2469	4.5868	
	23	0.625	126261	6	536	5825/211	7.082907	5.1013	4.8631	
	23	0.6875	172936	6	466	5825/211	7.464159	5.2379	5.1395	
	23	0.75	511397	6	647	5825/211	8.942515	5.7088	5.4159	
	23	0.8125	809884	6	600	5825/211	9.654663	5.9084	5.6922	
	23	0.875	814774	6	435	5825/211	9.664355	5.9110	5.9686	
	23	0.9375	814389	6	291	5825/211	9.663593	5.9108	6.2450	
	23	BFS	1004228	6	140		10.007034	6.0018		0.8788
16	23	0.5	1851	9	44	7275/117	2.306987	3.2674	2.7061	
	23	0.5625	1870	9	36	7275/117	2.309606	3.2718	3.1549	
	23	0.625	2504	9	35	7275/117	2.385756	3.3986	3.6038	
	23	0.6875	2551	9	29	7275/117	2.390691	3.4067	4.0526	
	23	0.75	22976	9	113	7275/117	3.052011	4.3613	4.5015	
	23	0.8125	43228	9	122	7275/117	3.274048	4.6358	4.9503	
	23	0.875	267829	9	283	7275/117	4.009547	5.4279	5.3992	
	23	0.9375	2798746	9	842	7275/117	5.203904	6.4470	5.8480	
	23	BFS	7270715	9	1104		5.786254	6.8616		0.8698
17	23	0.5	656	7	33	6501/102	2.525892	2.8169	2.3428	
	23	0.5625	665	7	26	6501/102	2.530814	2.8228	2.9162	
	23	0.625	711	7	21	6501/102	2.555112	2.8519	3.4895	
	23	0.6875	17143	7	192	6501/102	4.025961	4.2341	4.0629	
	23	0.75	28608	7	194	6501/102	4.331527	4.4565	4.6363	
	23	0.8125	190546	7	514	6501/102	5.679181	5.2800	5.2096	
	23	0.875	844063	7	813	6501/102	7.024655	5.9264	5.7830	
	23	0.9375	2558990	7	895	6501/102	8.23073	6.4081	6.3564	
	23	BFS	2749381	7	405		8.315545	6.4392		0.9498
18	23	0.5	683	8	14	6012/164	2.261011	2.8344	2.7250	
	23	0.5625	772	8	13	6012/164	2.295896	2.8876	3.3052	
	23	0.625	18190	8	114	6012/164	3.407839	4.2598	3.8855	
	23	0.6875	24869	8	107	6012/164	3.543703	4.3957	4.4657	
	23	0.75	136138	8	280	6012/164	4.382757	5.1340	5.0459	
	23	0.8125	308550	8	323	6012/164	4.854737	5.4893	5.6262	
	23	0.875	2311528	8	889	6012/164	6.244352	6.3639	6.2064	
	23	0.9375	4805568	8	1083	6012/164	6.842552	6.6817	6.7866	
	23	BFS	5201719	8	790		6.910641	6.7161		0.9729
19	23	0.5	2854	7	65	5503/119	3.116279	3.4555	3.2041	
	23	0.5625	3140	7	56	5503/119	3.159085	3.4969	3.6529	
	23	0.625	11500	7	121	5503/119	3.802767	4.0607	4.1017	
	23	0.6875	38170	7	210	5503/119	4.51369	4.5817	4.5505	
	23	0.75	51667	7	185	5503/119	4.713203	4.7132	4.9993	
	23	0.8125	270043	7	412	5503/119	5.969239	5.4314	5.4481	
	23	0.875	1107776	7	682	5503/119	7.302863	6.0445	5.8969	
	23	0.9375	2600747	7	772	5503/119	8.249784	6.4151	6.3457	
	23	BFS	2854529	7	415		8.360249	6.4555		0.9770
20	23	0.5	158012	7	866	6592/295	5.529298	5.1987	5.5119	
	23	0.5625	505837	7	1173	6592/295	6.52918	5.7040	5.5748	
	23	0.625	589456	7	965	6592/295	6.673447	5.7705	5.6376	
	23	0.6875	700571	7	797	6592/295	6.840134	5.8455	5.7005	
	23	0.75	682245	7	631	6592/295	6.814281	5.8339	5.7633	
	23	0.8125	682583	7	484	6592/295	6.814763	5.8342	5.8262	
	23	0.875	682855	7	357	6592/295	6.815151	5.8343	5.8890	
	23	0.9375	682455	7	235	6592/295	6.814581	5.8341	5.9518	
	23	BFS	906305	7	123		7.096418	5.9573		0.4860

Table 3: Results for the Knapsack instances of type Strongly Correlated.

#	n	Heuristic error δ	Total node expansions E	Optimal soln. depth d	Search time, seconds	$h^*([n])/m$	$\log_{10} E$	Linear fit to $\log_{10} E$	R^2
1	20	0.5	731425	11	1090	5509/28	5.8642	5.8687	
	20	0.5625	761013	15	878	5509/28	5.8814	5.8803	
	20	0.625	782339	12	716	5509/28	5.8934	5.8919	
	20	0.6875	805295	12	579	5509/28	5.9060	5.9036	
	20	0.75	828252	12	463	5509/28	5.9182	5.9152	
	20	0.8125	845545	10	360	5509/28	5.9271	5.9268	
	20	0.875	865626	11	267	5509/28	5.9373	5.9384	
	20	0.9375	885943	14	179	5509/28	5.9474	5.9500	
	20	BFS	900630	13	80		5.9545		0.9918
2	20	0.5	67164	6	259	2984/28	4.8271	4.8311	
	20	0.5625	71824	9	208	2984/28	4.8563	4.8558	
	20	0.625	76627	7	168	2984/28	4.8844	4.8804	
	20	0.6875	80614	8	136	2984/28	4.9064	4.9050	
	20	0.75	84553	8	107	2984/28	4.9271	4.9297	
	20	0.8125	90166	9	82	2984/28	4.9550	4.9543	
	20	0.875	96506	7	58	2984/28	4.9846	4.9790	
	20	0.9375	99536	7	35	2984/28	4.9980	5.0036	
	20	BFS	104144	8	10		5.0176		0.9959
3	20	0.5	222293	11	533	3687/26	5.3469	5.3552	
	20	0.5625	232989	12	432	3687/26	5.3673	5.3706	
	20	0.625	244871	8	353	3687/26	5.3889	5.3861	
	20	0.6875	256250	9	285	3687/26	5.4087	5.4015	
	20	0.75	266235	9	226	3687/26	5.4253	5.4170	
	20	0.8125	274056	8	173	3687/26	5.4378	5.4324	
	20	0.875	279890	11	126	3687/26	5.4470	5.4479	
	20	0.9375	283160	9	81	3687/26	5.4520	5.4633	
	20	BFS	291239	9	28		5.4642		0.9649
4	20	0.5	290608	10	329	3883/56	5.4633	5.4734	
	20	0.5625	304974	10	272	3883/56	5.4843	5.4834	
	20	0.625	313598	9	225	3883/56	5.4964	5.4935	
	20	0.6875	323477	9	185	3883/56	5.5098	5.5035	
	20	0.75	331235	9	151	3883/56	5.5201	5.5136	
	20	0.8125	336665	10	121	3883/56	5.5272	5.5237	
	20	0.875	340874	9	92	3883/56	5.5326	5.5337	
	20	0.9375	342644	9	64	3883/56	5.5348	5.5438	
	20	BFS	360837	8	33		5.5573		0.9369
5	20	0.5	851515	11	740	7731/77	5.9302	5.9348	
	20	0.5625	873968	14	609	7731/77	5.9415	5.9421	
	20	0.625	893378	12	498	7731/77	5.9510	5.9494	
	20	0.6875	912734	14	410	7731/77	5.9603	5.9567	
	20	0.75	927408	13	335	7731/77	5.9673	5.9641	
	20	0.8125	940724	12	267	7731/77	5.9735	5.9714	
	20	0.875	950209	12	206	7731/77	5.9778	5.9787	
	20	0.9375	958343	13	142	7731/77	5.9815	5.9860	
	20	BFS	967863	12	88		5.9858		0.9687
6	20	0.5	75858	10	488	2327/11	4.8800	4.8895	
	20	0.5625	81410	8	363	2327/11	4.9107	4.9155	
	20	0.625	88494	6	287	2327/11	4.9469	4.9416	
	20	0.6875	94585	9	225	2327/11	4.9758	4.9676	
	20	0.75	100329	7	177	2327/11	5.0014	4.9936	
	20	0.8125	106409	5	134	2327/11	5.0270	5.0197	
	20	0.875	110656	9	94	2327/11	5.0440	5.0457	
	20	0.9375	114601	9	55	2327/11	5.0592	5.0717	
	20	BFS	117496	4	11		5.0700		0.9833
7	20	0.5	712138	11	1178	6456/33	5.8526	5.8590	
	20	0.5625	748095	12	947	6456/33	5.8740	5.8727	
	20	0.625	778565	15	765	6456/33	5.8913	5.8864	
	20	0.6875	799378	11	618	6456/33	5.9028	5.9001	
	20	0.75	823236	13	490	6456/33	5.9155	5.9138	
	20	0.8125	844925	13	378	6456/33	5.9268	5.9275	
	20	0.875	870175	13	280	6456/33	5.9396	5.9412	
	20	0.9375	897407	12	185	6456/33	5.9530	5.9549	
	20	BFS	909075	14	80		5.9586		0.9895

Table 4: Results for the Knapsack instances of type Subset Sum.

#	n	Heuristic error δ	Total node expansions E	Optimal soln. depth d	Search time, seconds	$h^*([n])/m$	$\log_{10} E$	Linear fit to $\log_{10} E$	R^2
8	20	0.5	252054	10	2274	3514/7	5.4015	5.4113	
	20	0.5625	279643	12	1607	3514/7	5.4466	5.4416	
	20	0.625	299328	9	1159	3514/7	5.4761	5.4719	
	20	0.6875	324182	11	878	3514/7	5.5108	5.5023	
	20	0.75	340530	9	666	3514/7	5.5322	5.5326	
	20	0.8125	361756	10	494	3514/7	5.5584	5.5629	
	20	0.875	385942	10	344	3514/7	5.5865	5.5932	
	20	0.9375	423848	9	201	3514/7	5.6272	5.6236	
	20	BFS	454094	9	42		5.6571		0.9925
9	20	0.5	284146	9	628	4494/34	5.4535	5.4677	
	20	0.5625	301301	8	507	4494/34	5.4790	5.4812	
	20	0.625	318308	7	412	4494/34	5.5028	5.4947	
	20	0.6875	330924	9	334	4494/34	5.5197	5.5083	
	20	0.75	338590	9	263	4494/34	5.5297	5.5218	
	20	0.8125	345335	9	203	4494/34	5.5382	5.5353	
	20	0.875	351027	10	146	4494/34	5.5453	5.5489	
	20	0.9375	356374	10	92	4494/34	5.5519	5.5624	
	20	BFS	369094	8	34		5.5671		0.9282
10	20	0.5	812828	11	1078	6963/39	5.9100	5.9193	
	20	0.5625	852539	13	874	6963/39	5.9307	5.9298	
	20	0.625	881657	15	711	6963/39	5.9453	5.9403	
	20	0.6875	903389	12	579	6963/39	5.9559	5.9508	
	20	0.75	923450	15	466	6963/39	5.9654	5.9613	
	20	0.8125	941277	11	356	6963/39	5.9737	5.9717	
	20	0.875	954861	14	266	6963/39	5.9799	5.9822	
	20	0.9375	970871	14	180	6963/39	5.9872	5.9927	
	20	BFS	985526	12	88		5.9937		0.9593
11	20	0.5	872387	12	527	7270/102	5.9407	5.9456	
	20	0.5625	892404	13	441	7270/102	5.9506	5.9507	
	20	0.625	907719	12	366	7270/102	5.9580	5.9558	
	20	0.6875	920529	12	306	7270/102	5.9640	5.9609	
	20	0.75	930373	12	260	7270/102	5.9687	5.9660	
	20	0.8125	939495	13	214	7270/102	5.9729	5.9711	
	20	0.875	945766	12	169	7270/102	5.9758	5.9762	
	20	0.9375	948094	11	125	7270/102	5.9769	5.9813	
	20	BFS	961185	11	85		5.9828		0.9409
12	20	0.5	544749	13	997	5752/35	5.7362	5.7422	
	20	0.5625	572592	8	804	5752/35	5.7578	5.7579	
	20	0.625	596732	12	656	5752/35	5.7758	5.7736	
	20	0.6875	622826	9	528	5752/35	5.7944	5.7893	
	20	0.75	644836	11	420	5752/35	5.8094	5.8050	
	20	0.8125	662145	12	329	5752/35	5.8210	5.8207	
	20	0.875	682257	11	242	5752/35	5.8339	5.8364	
	20	0.9375	705866	11	158	5752/35	5.8487	5.8521	
	20	BFS	720827	13	64		5.8578		0.9901
13	20	0.5	592766	10	1824	7445/30	5.7729	5.7767	
	20	0.5625	628513	10	1319	7445/30	5.7983	5.7963	
	20	0.625	662306	11	1040	7445/30	5.8211	5.8159	
	20	0.6875	684651	13	828	7445/30	5.8355	5.8355	
	20	0.75	713728	12	645	7445/30	5.8535	5.8552	
	20	0.8125	745263	10	487	7445/30	5.8723	5.8748	
	20	0.875	781953	11	344	7445/30	5.8932	5.8944	
	20	0.9375	824260	11	216	7445/30	5.9161	5.9140	
	20	BFS	861415	11	74		5.9352		0.9963
14	20	0.5	137368	8	561	3509/22	5.1379	5.1513	
	20	0.5625	148933	7	450	3509/22	5.1730	5.1713	
	20	0.625	157793	10	363	3509/22	5.1981	5.1913	
	20	0.6875	165368	9	289	3509/22	5.2185	5.2113	
	20	0.75	172383	9	226	3509/22	5.2365	5.2314	
	20	0.8125	179983	7	173	3509/22	5.2552	5.2514	
	20	0.875	186068	9	123	3509/22	5.2697	5.2714	
	20	0.9375	191426	8	73	3509/22	5.2820	5.2914	
	20	BFS	197634	10	18		5.2959		0.9761

Table 5: Results for the Knapsack instances of type Subset Sum.

#	n	Heuristic error δ	Total node expansions E	Optimal soln. depth d	Search time, seconds	$h^*([n])/m$	$\log_{10} E$	Linear fit to $\log_{10} E$	R^2
15	20	0.5	34937	9	1022	3124/9	4.5433	4.5311	
	20	0.5625	38617	6	772	3124/9	4.5868	4.5760	
	20	0.625	41757	10	529	3124/9	4.6207	4.6209	
	20	0.6875	45036	9	353	3124/9	4.6536	4.6658	
	20	0.75	49231	10	272	3124/9	4.6922	4.7107	
	20	0.8125	54428	7	186	3124/9	4.7358	4.7556	
	20	0.875	62409	9	128	3124/9	4.7952	4.8006	
	20	0.9375	75602	7	72	3124/9	4.8785	4.8455	
	20	BFS	84284	8	8		4.9257		0.9739
16	20	0.5	476547	10	3224	5442/11	5.6781	5.6718	
	20	0.5625	498939	11	2097	5442/11	5.6980	5.6976	
	20	0.625	523867	10	1536	5442/11	5.7192	5.7235	
	20	0.6875	558927	10	1181	5442/11	5.7474	5.7493	
	20	0.75	592373	9	911	5442/11	5.7726	5.7751	
	20	0.8125	626403	10	675	5442/11	5.7969	5.8010	
	20	0.875	668497	12	468	5442/11	5.8251	5.8268	
	20	0.9375	725325	12	281	5442/11	5.8605	5.8527	
	20	BFS	768536	13	71		5.8857		0.9947
17	20	0.5	641544	15	3751	7157/11	5.8072	5.8045	
	20	0.5625	666837	11	2791	7157/11	5.8240	5.8256	
	20	0.625	702032	12	1991	7157/11	5.8464	5.8468	
	20	0.6875	737893	14	1495	7157/11	5.8680	5.8679	
	20	0.75	772405	14	1124	7157/11	5.8878	5.8891	
	20	0.8125	810089	14	827	7157/11	5.9085	5.9102	
	20	0.875	852271	14	570	7157/11	5.9306	5.9313	
	20	0.9375	902227	12	337	7157/11	5.9553	5.9525	
	20	BFS	964897	14	86		5.9845		0.9988
18	20	0.5	321490	9	1215	4631/20	5.5072	5.5047	
	20	0.5625	338267	10	952	4631/20	5.5293	5.5293	
	20	0.625	358571	9	760	4631/20	5.5546	5.5540	
	20	0.6875	379827	10	600	4631/20	5.5796	5.5786	
	20	0.75	399061	9	466	4631/20	5.6010	5.6033	
	20	0.8125	419052	10	356	4631/20	5.6223	5.6279	
	20	0.875	443204	9	252	4631/20	5.6466	5.6525	
	20	0.9375	486366	10	157	4631/20	5.6870	5.6772	
	20	BFS	508524	10	47		5.7063		0.9932
19	20	0.5	104698	7	251	3373/44	5.0199	5.0322	
	20	0.5625	110845	8	206	3373/44	5.0447	5.0482	
	20	0.625	116893	10	169	3373/44	5.0678	5.0641	
	20	0.6875	122710	8	137	3373/44	5.0889	5.0800	
	20	0.75	128398	6	110	3373/44	5.1086	5.0959	
	20	0.8125	131887	9	84	3373/44	5.1202	5.1119	
	20	0.875	133658	10	60	3373/44	5.1260	5.1278	
	20	0.9375	134205	5	37	3373/44	5.1278	5.1437	
	20	BFS	142348	8	13		5.1534		0.9349
20	20	0.5	275501	10	352	5262/94	5.4401	5.4489	
	20	0.5625	286961	9	292	5262/94	5.4578	5.4594	
	20	0.625	296924	9	240	5262/94	5.4726	5.4699	
	20	0.6875	305914	7	196	5262/94	5.4856	5.4804	
	20	0.75	315286	9	159	5262/94	5.4987	5.4909	
	20	0.8125	322234	8	126	5262/94	5.5082	5.5013	
	20	0.875	324077	9	94	5262/94	5.5106	5.5118	
	20	0.9375	324471	10	65	5262/94	5.5112	5.5223	
	20	BFS	348398	9	32		5.5421		0.9299

Table 6: Results for the Knapsack instances of type Subset Sum.

#	n	k	Heuristic error δ	Total node expansions E	Effective branching factor $E^{1/k}$	Upper bound $B(d)^{1/k}$	$\frac{B(d)^{1/k}}{E^{1/k}}$	$\log_{10} E$	Linear fit to $\log_{10} E$	R^2
1	10	44	0	87	1.10682761	1.12498287	1.01640297	1.9395	1.2674	
	10	44	0.0025	87	1.10682761	1.12509476	1.01650406	1.9395	1.3758	
	10	44	0.005	87	1.10682761	1.12524953	1.01664390	1.9395	1.4843	
	10	44	0.0075	87	1.10682761	1.12546320	1.01683694	1.9395	1.5928	
	10	44	0.01	87	1.10682761	1.12575740	1.01710275	1.9395	1.7013	
	10	44	0.0125	87	1.10682761	1.12616102	1.01746741	1.9395	1.8097	
	10	44	0.015	87	1.10682761	1.12671203	1.01796524	1.9395	1.9182	
	10	44	0.0175	87	1.10682761	1.12745936	1.01864044	1.9395	2.0267	
	10	44	0.02	87	1.10682761	1.12846421	1.01954830	1.9395	2.1352	
	10	44	0.0225	87	1.10682761	1.12980027	1.02075541	1.9395	2.2436	
	10	44	0.025	135	1.11793532	1.29413023	1.15760743	2.1303	2.3521	
	10	44	0.0275	177	1.12483883	1.29413756	1.15050932	2.2480	2.4606	
	10	44	0.03	219	1.13029527	1.29414775	1.14496431	2.3404	2.5691	
	10	44	0.0325	261	1.13481129	1.29416190	1.14042036	2.4166	2.6775	
	10	44	0.035	289	1.13744262	1.29418158	1.13779944	2.4609	2.7860	
	10	44	0.0375	317	1.13983570	1.29420890	1.13543461	2.5011	2.8945	
	10	44	0.04	345	1.14203051	1.29424685	1.13328571	2.5378	3.0030	
	10	44	0.0425	359	1.14306342	1.29429954	1.13230772	2.5551	3.1114	
	10	44	0.045	373	1.14405770	1.29437264	1.13138755	2.5717	3.2199	
	10	44	0.0475	531	1.15327789	1.48549510	1.28806345	2.7251	3.3284	
	10	44	0.05	2530	1.19493326	1.48549548	1.24316189	3.4031	3.4369	
	10	44	0.0525	3458	1.20344942	1.48549601	1.23436514	3.5388	3.5454	
	10	44	0.055	5709	1.21724042	1.48549674	1.22038072	3.7566	3.6538	
	10	44	0.0575	8539	1.22842928	1.48549775	1.20926599	3.9314	3.7623	
	10	44	0.06	10183	1.23335496	1.48549917	1.20443766	4.0079	3.8708	
	10	44	0.0625	13956	1.24222170	1.48550113	1.19584220	4.1448	3.9793	
	10	44	0.065	16041	1.24615895	1.48550386	1.19206612	4.2052	4.0877	
	10	44	0.0675	18293	1.24988516	1.48550766	1.18851532	4.2623	4.1962	
	10	44	0.07	23400	1.25689894	1.68167021	1.33795181	4.3692	4.3047	
	10	44	0.0725	33251	1.26697571	1.68167024	1.32731056	4.5218	4.4132	
	10	44	0.075	54989	1.28154406	1.68167029	1.31222199	4.7403	4.5216	
	10	44	0.0775	69492	1.28838000	1.68167036	1.30525960	4.8419	4.6301	
	10	44	0.08	85507	1.29446689	1.68167046	1.29912203	4.9320	4.7386	
	10	44	0.0825	99904	1.29905304	1.68167059	1.29453574	4.9996	4.8471	
	10	44	0.085	118924	1.30420852	1.68167077	1.28941863	5.0753	4.9555	
	10	44	0.0875	139520	1.30895150	1.68167103	1.28474663	5.1446	5.0640	
	10	44	0.09	158117	1.31267920	1.68167138	1.28109852	5.1990	5.1725	
	10	44	0.0925	181666	1.31682768	1.88726770	1.43319261	5.2593	5.2810	
	10	44	0.095	258998	1.32748452	1.88726771	1.42168717	5.4133	5.3894	
	10	44	0.0975	475269	1.34592652	1.88726771	1.40220709	5.6769	5.4979	
										0.9482
2	12	63	0	125	1.07965322	1.09187259	1.01131786	2.0969	1.3953	
	12	63	0.0025	125	1.07965322	1.09196102	1.01139977	2.0969	1.5797	
	12	63	0.005	125	1.07965322	1.09210593	1.01153399	2.0969	1.7641	
	12	63	0.0075	125	1.07965322	1.09234240	1.01175301	2.0969	1.9485	
	12	63	0.01	125	1.07965322	1.09272569	1.01210802	2.0969	2.1328	
	12	63	0.0125	125	1.07965322	1.09334029	1.01267728	2.0969	2.3172	
	12	63	0.015	125	1.07965322	1.09430956	1.01357504	2.0969	2.5016	
	12	63	0.0175	295	1.09446915	1.21404534	1.10925497	2.4698	2.6860	
	12	63	0.02	599	1.10684330	1.21404945	1.09685756	2.7774	2.8704	
	12	63	0.0225	789	1.11169422	1.21405622	1.09207747	2.8971	3.0547	
	12	63	0.025	979	1.11550813	1.21406738	1.08835368	2.9908	3.2391	
	12	63	0.0275	1093	1.11746021	1.21408579	1.08646892	3.0386	3.4235	
	12	63	0.03	1207	1.11922136	1.21411611	1.08478640	3.0817	3.6079	
	12	63	0.0325	1759	1.12593198	1.34843434	1.19761616	3.2453	3.7923	
	12	63	0.035	8006	1.15334431	1.34843446	1.16915170	3.9034	3.9767	
	12	63	0.0375	18159	1.16843520	1.34843466	1.15405173	4.2591	4.1610	
	12	63	0.04	31829	1.17889026	1.34843500	1.14381723	4.5028	4.3454	
	12	63	0.0425	39898	1.18312592	1.34843555	1.13972277	4.6010	4.5298	
	12	63	0.045	53605	1.18868491	1.34843647	1.13439352	4.7292	4.7142	
	12	63	0.0475	63934	1.19201428	1.34843797	1.13122636	4.8057	4.8986	
	12	63	0.05	151470	1.20844644	1.48271141	1.22695666	5.1803	5.0829	
	12	63	0.0525	240217	1.21732463	1.48271142	1.21800824	5.3806	5.2673	
	12	63	0.055	418262	1.22808758	1.48271144	1.20733363	5.6214	5.4517	
	12	63	0.0575	569663	1.23412462	1.48271147	1.20142768	5.7556	5.6361	
	12	63	0.06	823942	1.24137536	1.48271152	1.19441030	5.9159	5.8205	
	12	63	0.0625	1.03E+06	1.24580697	1.48271161	1.19016159	6.0134	6.0049	
	12	63	0.065	1.39E+06	1.25172483	1.62031036	1.29446211	6.1431	6.1892	
	12	63	0.0675	3.35E+06	1.26929396	1.62031036	1.27654461	6.5244	6.3736	
	12	63	0.07	6.43E+06	1.28251719	1.62031037	1.26338296	6.8080	6.5580	
										0.9693

Table 7: Results for partial Latin square instances.

#	n	k	Heuristic error δ	Total node expansions E	Effective branching factor $E^{1/k}$	Upper bound $B(d)^{1/k}$	$\frac{B(d)^{1/k}}{E^{1/k}}$	$\log_{10} E$	Linear fit to $\log_{10} E$	R^2
3	14	86	0	171	1.06161017	1.07034588	1.00822873	2.2330	1.4986	
	14	86	0.0025	171	1.06161017	1.07042098	1.00829948	2.2330	1.7445	
	14	86	0.005	171	1.06161017	1.07057335	1.00844300	2.2330	1.9904	
	14	86	0.0075	171	1.06161017	1.07087962	1.00873150	2.2330	2.2363	
	14	86	0.01	171	1.06161017	1.07148429	1.00930108	2.2330	2.4822	
	14	86	0.0125	247	1.06615920	1.16291112	1.09074810	2.3927	2.7281	
	14	86	0.015	429	1.07302532	1.16291347	1.08377077	2.6325	2.9740	
	14	86	0.0175	555	1.07624311	1.16291827	1.08053493	2.7443	3.2199	
	14	86	0.02	667	1.07854600	1.16292811	1.07823691	2.8241	3.4658	
	14	86	0.0225	737	1.07979831	1.16294821	1.07700503	2.8675	3.7117	
	14	86	0.025	6959	1.10835983	1.26274158	1.13928848	3.8425	3.9576	
	14	86	0.0275	27506	1.12621485	1.26274166	1.12122626	4.4394	4.2035	
	14	86	0.03	57104	1.13582148	1.26274182	1.11174321	4.7567	4.4494	
	14	86	0.0325	90923	1.14198131	1.26274214	1.10574676	4.9587	4.6953	
	14	86	0.035	122879	1.14598774	1.36083647	1.18747908	5.0895	4.9412	
	14	86	0.0375	259053	1.15596951	1.36083648	1.17722523	5.4134	5.1871	
	14	86	0.04	463344	1.16381138	1.36083648	1.16929298	5.6659	5.4330	
	14	86	0.0425	665871	1.16872905	1.36083649	1.16437295	5.8234	5.6789	
	14	86	0.045	925306	1.17320907	1.36083651	1.15992669	5.9663	5.9248	
	14	86	0.0475	1.29E+06	1.17776024	1.45985179	1.23951526	6.1109	6.1707	
										0.9335
4	16	113	0	225	1.04909731	1.05563497	1.00623170	2.3522	1.6772	
	16	113	0.0025	225	1.04909731	1.05570312	1.00629666	2.3522	2.0340	
	16	113	0.005	225	1.04909731	1.05588217	1.00646733	2.3522	2.3907	
	16	113	0.0075	225	1.04909731	1.05634285	1.00690645	2.3522	2.7475	
	16	113	0.01	799	1.06092884	1.12838087	1.06357828	2.9025	3.1042	
	16	113	0.0125	1719	1.06814635	1.12838284	1.05639348	3.2353	3.4610	
	16	113	0.015	2317	1.07097198	1.12838808	1.05361121	3.3649	3.8178	
	16	113	0.0175	2731	1.07253118	1.12840202	1.05209251	3.4363	4.1745	
	16	113	0.02	50236	1.10053004	1.20572650	1.09558708	4.7010	4.5313	
	16	113	0.0225	144797	1.11088842	1.20572656	1.08537144	5.1608	4.8881	
	16	113	0.025	258735	1.11660964	1.20572671	1.07981041	5.4129	5.2448	
	16	113	0.0275	516942	1.12346988	1.28088203	1.14011248	5.7134	5.6016	
	16	113	0.03	1.97E+06	1.13686805	1.28088203	1.12667607	6.2952	5.9584	
										0.9274
5	18	143	0	285	1.04031952	1.04542550	1.00490809	2.4548	1.8665	
	18	143	0.0025	285	1.04031952	1.04549145	1.00497148	2.4548	2.3144	
	18	143	0.005	285	1.04031952	1.04572413	1.00519515	2.4548	2.7623	
	18	143	0.0075	743	1.04731385	1.10463010	1.05472691	2.8710	3.2103	
	18	143	0.01	2579	1.05646789	1.10463134	1.04558912	3.4115	3.6582	
	18	143	0.0125	3659	1.05905525	1.10463580	1.04303887	3.5634	4.1061	
	18	143	0.015	39137	1.07675277	1.16693654	1.08375532	4.5926	4.5540	
	18	143	0.0175	246338	1.09069423	1.16693656	1.06990257	5.3915	5.0019	
	18	143	0.02	535932	1.09663904	1.16693665	1.06410278	5.7291	5.4498	
										0.9120
6	20	176	0	351	1.03386057	1.03797380	1.00397851	2.5453	1.9462	
	20	176	0.0025	351	1.03386057	1.03804139	1.00404389	2.5453	2.5098	
	20	176	0.005	351	1.03386057	1.03837263	1.00436428	2.5453	3.0733	
	20	176	0.0075	2425	1.04527681	1.08739144	1.04029040	3.3847	3.6368	
	20	176	0.01	4125	1.04843662	1.08739402	1.03715761	3.6154	4.2004	
	20	176	0.0125	107153	1.06802045	1.13899860	1.06645766	5.0300	4.7639	
	20	176	0.015	619190	1.07871841	1.13899862	1.05588132	5.7918	5.3275	
										0.8698

Table 8: Results for partial Latin square instances.

References

Aaronson, S. (2004). Lower bounds for local search by quantum arguments. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*. ACM Press.

Babai, L. (1991). Local expansion of vertex-transitive graphs and random generation in finite groups. In *Proceedings of the 23rd annual ACM symposium on Symposium of Theory of Computing*, pp. 164–174.

Chernoff, H. (1952). A measure of asymptotic efficiency for tests of hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, *23*, 493–507.

Chung, F. (2006). The diameter and Laplacian eigenvalues of directed graphs. *Electronic Journal of Combinatorics*, *13*(4).

Chung, F. R. K. (1997). *Spectral Graph Theory*. American Mathematical Society.

Colbourn, C. J. (1984). The complexity of completing partial Latin squares. *Discrete Applied Mathematics*, *8*(1), 25–30.

Davis, H., Bramanti-Gregor, A., & Wang, J. (1988). The advantages of using depth and breadth components in heuristic search. In Ras, Z., & Saitta, L. (Eds.), *Proceedings of the Third International Symposium on Methodologies for Intelligent Systems*, pp. 19–28, North-Holland, Amsterdam. Elsevier.

Dechter, R., & Pearl, J. (1985). Generalized best-first search strategies and the optimality of A*. *J. ACM*, *32*(3), 505–536.

Demaine, E. D. (2001). Playing games with algorithms: Algorithmic combinatorial game theory. In *Proc. 26th Symp. on Math Found. in Comp. Sci., Lect. Notes in Comp. Sci.*, pp. 18–32. Springer-Verlag.

Dinh, H., Russell, A., & Su, Y. (2007). On the value of good advice: The complexity of A* with accurate heuristics. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*, pp. 1140–1145.

Edelkamp, S. (2001). Prediction of regular search tree growth by spectral analysis. In *Proceedings of the Joint German/Austrian Conference on AI: Advances in Artificial Intelligence*, KI '01, pp. 154–168, London, UK, UK. Springer-Verlag.

Felner, A., Korf, R. E., & Hanan, S. (2004). Additive pattern database heuristics. *Journal of Artificial Intelligence Research*, *22*, 279–318.

Friedman, J. (2003). A proof of Alon's second eigenvalue conjecture. In *STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pp. 720–724, New York, NY, USA. ACM.

Gaschnig, J. (1979). *Perfomance measurement and analysis of certain search algorithms*. Ph.D. thesis, Carnegie-Mellon University, Pittsburgh, PA.

Gomes, C., & Shmoys, D. (2002). Completing quasigroups or Latin squares: A structured graph coloring problem. In Johnson, D. S., Mehrotra, A., & Trick, M. (Eds.), *Proceedings of the Computational Symposium on Graph Coloring and its Generalizations*, pp. 22–39, Ithaca, New York, USA.

Gomes, C. P., Selman, B., & Kautz, H. (1998). Boosting combinatorial search through randomization. In *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pp. 431–437, Menlo Park, CA, USA. American Association for Artificial Intelligence.

Hart, P., Nilson, N., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, *SCC-4*(2), 100–107.

Helmert, M., & Röger, G. (2008). How good is almost perfect?. In *Proceedings of AAAI-08*.

Hochbaum, D. (1996). *Approximation Algorithms for NP-hard Problems*. Brooks Cole.

Horn, R., & Johnson, C. (1999). *Matrix Analysis*. Cambridge University Press, Cambridge, UK.

Huyn, N., Dechter, R., & Pearl, J. (1980). Probabilistic analysis of the complexity of A*. *Artificial Intelligence*, *15*, 241–254.

Ibarra, O. H., & Kim, C. E. (1975). Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, *22*(4), 463–468.

Karp, R. M. (1972). Reducibility among combinatorial problems. In Miller, R. E., & Thatcher, J. W. (Eds.), *Complexity of Computer Computations*, p. 85103. New York: Plenum.

Kellerer, H., Pferschy, U., & Pisinger, D. (2004). *Knapsack problems*. Springer.

Korf, R. (1985). Depth-first iterative deepening: An optimal admissible tree search. *Artificial Intelligence*, *27*, 97–109.

Korf, R., & Reid, M. (1998). Complexity analysis of admissible heuristic search. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-98)*, pp. 305–310.

Korf, R., Reid, M., & Edelkamp, S. (2001). Time complexity of iterative-deepening-A*. *Artificial Intelligence*, *129*(1-2), 199–218.

Korf, R. E. (2000). Recent progress in the design and analysis of admissible heuristic functions. In *AAAI/IAAI 2000*, pp. 1165–1170. Also in SARA '02: Proceedings of the 4th International Symposium on Abstraction, Reformulation, and Approximation.

Kumar, R., Russell, A., & Sundaram, R. (1996). Approximating Latin square extensions. In *COCOON '96: Proceedings of the Second Annual International Conference on Computing and Combinatorics*, pp. 280–289, London, UK. Springer-Verlag.

Laywine, C., & Mullen, G. (1998). *Discrete Mathematics using Latin Squares*. Interscience Series in Discrete mathematics and Optimization. Wiley.

Lenstra, A. K., Lenstra, H. W., & Lovasz, L. (1981). Factoring polynomials with rational coefficients. Tech. rep. 82-05, Universiteit Amsterdam.

Motwani, R., & Raghavan, P. (1995). *Randomized Algorithms*. Cambridge University Press.

Parberry, I. (1995). A real-time algorithm for the (n^2-1)-puzzle. *Inf. Process. Lett*, *56*, 23–28.

Pearl, J. (1984). *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, MA.

Pisinger, D. (2005). Where are the hard knapsack problems?. *Computers and Operations Research*, *32*, 2271–2284.

Pohl, I. (1977). Practical and theoretical considerations in heuristic search algorithms. In Elcock, W., & Michie, D. (Eds.), *Machine Intelligence*, Vol. 8, pp. 55–72. Ellis Horwood, Chichester.

Ratner, D., & Warmuth, M. (1990). The (n^2-1)-puzzle and related relocation problems. *Journal for Symbolic Computation*, *10*(2), 111–137.

Russell, S., & Norvig, P. (1995). *Artificial Intelligence - A Modern Approach*. Prentice Hall, New Jersey.

Sen, A. K., Bagchi, A., & Zhang, W. (2004). Average-case analysis of best-first search in two representative directed acyclic graphs. *Artif. Intell.*, *155*(1-2), 183–206.

Tay, T.-S. (1996). Some results on generalized Latin squares. *Graphs and Combinatorics*, *12*, 199–207.

Vazirani, V. (2001). *Approximation Algorithms*. Springer-Verlag.

Vazirani, V. (2002). Primal-dual schema based approximation algorithms. In *Theoretical Aspects of Computer Science: Advanced Lectures*, pp. 198–207. Springer-Verlag, New York.

Zahavi, U., Felner, A., Schaeffer, J., & Sturtevant, N. (2007). Inconsistent heuristics. In *Proceedings of AAAI-07*, pp. 1211–1216.

Zhang, Z., Sturtevant, N. R., Holte, R., Schaeffer, J., & Felner, A. (2009). A* search with inconsistent heuristics. In *Proceedings of the 21st international jont conference on Artifical intelligence*, IJCAI'09, pp. 634–639, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.