Simultaneous Heuristic Search for Conjunctive Subgoals

Lin Zhu and **Robert Givan**
Electrical and Computer Engineering, Purdue University, West Lafayette IN 47907 USA
{lzhu, givan}@purdue.edu

Abstract

We study the problem of building effective heuristics for achieving conjunctive goals from heuristics for individual goals. We consider a straightforward method for building conjunctive heuristics that smoothly trades off between previous common methods. In addition to first explicitly formulating the problem of designing conjunctive heuristics, our major contribution is the discovery that this straightforward method substantially outperforms previously used methods across a wide range of domains. Based on a single positive real parameter k, our heuristic measure sums the individual heuristic values for the subgoal conjuncts, each raised to the k'th power. Varying k allows loose approximation and combination of the previous min, max, and sum approaches, while mitigating some of the weaknesses in those approaches. Our empirical work shows that for many benchmark planning domains there exist fixed parameter values that perform well—we give evidence that these values can be found automatically by training. Our method, applied to top-level conjunctive goals, shows dramatic improvements over the heuristic used in the FF planner across a wide range of planning competition benchmarks. Also, our heuristic, without computing landmarks, consistently improves upon the success ratio of a recently published landmark-based planner FF-L.

Introduction

Good heuristic functions have been an important factor for success in many modern planners (Bonet & Geffner 2001; Hoffmann & Nebel 2001; Gerevini, Saetti, & Serina 2003; Helmert 2004). Here, we consider the problem of building a *conjunctive heuristic*: a heuristic function for conjunctive goals, given heuristics for the conjuncts (subgoals).

The need to seek a conjunction of subgoals occurs in many contexts in planning. Typically, planning goals are directly represented as a conjunctive set of literals. Furthermore, in partial-order planning, a given partial plan implicitly represents the conjunction of the subgoals of resolving all *threats* and supporting all *open conditions* (McAllester & Rosenblitt 1991). Other regression approaches, such as the backward phase of GRAPHPLAN (Blum & Furst 1997), maintain at any time a conjunctive set of open subgoals.

For a further example of conjunctive goals, we consider *pattern databases* (Culberson & Schaeffer 1996), a proven successful source of heuristics in recent years from the search community. A pattern database is a pre-computed lookup table of solution lengths for a relaxation of the problem domain. Multiple pattern databases, i.e., multiple different relaxations, are typically combined to form a single heuristic that seeks to drive each database heuristic to zero, often by computing the maximum or sum over the available pattern databases (Korf 2000). We view the resulting heuristic as seeking a conjunction of subgoals, each stating that a pattern database heuristic is zero. Edelkamp (2002) used automatically constructed pattern databases in planning.

A final example arises with planning *landmarks*. Hoffmann et al. (2004) introduced landmarks: these are propositions that every correct solution achieves (makes true) at some point during execution. Planning goals can be extended usefully by adding a conjunct for each landmark asserting that that landmark has been achieved, leading again to conjunctive goals.

Most planners in the literature can be viewed as constructing a conjunctive heuristic, either implicitly or explicitly. In previous work, a conjunctive heuristic is often computed as either the sum (Korf 2000; Bonet & Geffner 2001; Hoffmann & Nebel 2001; Nguyen & Kambhampati 2001), the minimum (Hoffmann, Porteous, & Sebastia 2004), or the maximum (Haslum & Geffner 2000; Korf 2000; Edelkamp 2002) of the subgoal heuristics[1]. These methods do not generally accurately reflect interactions between subgoals. In consequence, the resulting conjunctive heuristic may be misleading even if true distance is provided directly as the subgoal heuristic. For instance, the sum of the subgoal heuristics gives a large plateau in many transportation domains, where approaching one subgoal (e.g., city) requires simultaneously moving away from another. Here, negative subgoal interaction is being ignored in the summation. Even so, each of these simple conjunctive heuristics performs well in some subset of the standard planning benchmark domains.

We consider a straightforward method for building conjunctive heuristics that smoothly trades off between the previous methods. In addition to first explicitly formulating the problem of designing conjunctive heuristics, our major

[1]The use of the minimum subgoal heuristic requires combination with the number of unachieved subgoals as discussed in the next section.

contribution here is the discovery that this straightforward method substantially outperforms previously used methods across a wide range of domains. Based on a single, positive real parameter k, our heuristic measure sums the individual subgoal heuristics, each raised to the k'th power. Different values of k allow the loose approximation of each previous method (sum, min, or max) discussed above, and intermediate values of k select a tradeoff between these methods: for example, focusing on the closest subgoal, but not without regard for the effect on the further subgoals. We will demonstrate empirically that intermediate values of k can be very important to performance, and that the "loose approximations" of sum, min, or max typically outperform sum, min, or max, respectively for subtle reasons discussed below. We expect this discovery to be relevant to performance improvement for a wide range of planners.

Our experimental results show that, while the ideal value of k varies from domain to domain, for typical benchmark planning domains there exist fixed parameter values that perform well—we give evidence below that these values can then be found automatically by training on the domain.

Our method significantly improves FF's performance on a set of hard domains (Hoffmann 2002) for FF, as well as on a representative search domain, the 24-puzzle, while retaining FF's performance in other domains. We also show, strikingly, that the use of our heuristic, without computing or using landmarks, consistently improves upon the success ratio of a recently published planner FF-L that computes and exploits landmarks (Hoffmann, Porteous, & Sebastia 2004), on the domains that FF-L was demonstrated on.

In the following sections, we will introduce our method, compare it qualitatively to previous methods, and show a broad range of empirical results.

Heuristics for conjunctive goals

A heuristic function is a function mapping state-goal pairs to a totally ordered set with minimal and maximal elements $\bot$ and $\top$. This totally ordered set is typically the non-negative real numbers together with ∞. We assume throughout that a heuristic $h(s, g)$ is $\bot$ if and only if the goal g is achieved in state s, and $\top$ only if g is unreachable from s. We will sometimes omit the goal argument g to a heuristic function when it is assumed known implicitly.

The problem we consider is to build an effective conjunctive heuristic $H(s, G)$ for a conjunctive goal G, given a vector of heuristics $h(s, g_i)$ for each subgoal $g_i \in G$.

Here, we focus on the total quasi-order on states[2] induced by heuristic functions of interest. Recent previous work (Bonet & Geffner 2001; Hoffmann & Nebel 2001) has also primarily exploited the ordering information in heuristic functions. This limited focus on ordering derives from a best-first or greedy goal-seeking approach characteristic of sub-optimal planning—for optimal planning, an A*-like quantitative use of the exact heuristic values is suggested. Given an implicit, fixed goal g, we use $\preceq_h$ to represent the quasi-ordering *induced* by h:

$$s_1 \preceq_h s_2 \text{ iff } h(s_1) \leq h(s_2).$$

Several commonly considered conjunctive heuristics are sum, max, count, and min:

$$H_{\text{sum}}(s, G) = \sum_i h(s, g_i)$$
$$H_{\text{max}}(s, G) = \max_i h(s, g_i)$$

$H_{\text{count}}(s, G)$ and $H_{\text{min}}(s, G)$ are best defined in words. $H_{\text{count}}(s, G)$ is simply the number of unachieved subgoals in G, as long as they are all reachable, and ∞ otherwise. Informally, H_{min} then returns this count, breaking ties with the distance to the closest subgoal[3].

We argue below that these methods are often unsuitable even if true distance is provided directly for each subgoal heuristic. Here, we mix these methods by applying a parameterized nonlinear transformation on the subgoal heuristic values before the summation. The transformation we explore here is simply to raise the subgoal heuristic values to the k'th power, for a positive real parameter k. So, we define

$$H_k(s, G) = \sum_i h(s, g_i)^k$$

By varying k, the heuristic H_k can loosely resemble each of the basic heuristics H_{sum}, H_{max}, and H_{min}. For large values of k, $H_k(G)$ is most sensitive to the heuristic value of the furthest goal g_i, like H_{max}. When k is one, $H_k(G)$ equals H_{sum}. Finally, as k approaches zero, $H_k(G)$ is most sensitive to the reversal of achieved subgoals, and then to the heuristic value of the closest unachieved subgoal[4], just as is H_{min}. Intermediate values of k give behaviors combining the properties of the above heuristics. We will discuss the approximation of sum, min, and max by H_k further below.

Before presenting experimental results for the use of H_k, we first discuss the qualitative strengths and weaknesses of each basic conjunctive heuristic, suggesting reasons why a tradeoff heuristic like H_k may outperform each basic heuristic. We also mention key planning systems that employ variants of each basic heuristic.

Combination via SUM

Perhaps the most popular conjunctive heuristic is H_{sum}, as used, for example, in REPOP (Nguyen & Kambhampati 2001) and in HSP (Bonet & Geffner 2001) when selecting one frequently successful setting.

The heuristic used in the very successful planner FF (Hoffmann & Nebel 2001) counts the steps in a plan

[2]We use the term *total quasi-order* for a total, reflexive, transitive relation—such a relation totally orders equivalence classes of its domain elements.

[3]More formally, $H_{\text{min}}(G)$ returns values in $\Re \times \Re$, which are then totally ordered lexicographically. $H_{\text{min}}(G)$ returns the pair of $H_{\text{count}}(G)$ and the smallest non-zero subgoal distance. This lexicographic ordering approach is necessary to reward actions that achieve subgoals even though they also force the heuristic to start focusing on a further subgoal.

[4]We provide a minimal example to clarify the behavior of H_k as k approaches zero. Let (i, j) represent a state with an ordered pair of subgoal distances i and j. Both H_{min} and $H_{0.5}$ would order the following states in the same sequence: $(0, 5), (1, 4), (2, 3)$.

that achieves the conjunctive goal, ignoring delete lists of all actions. Because steps may be shared in plans for different subgoals, the FF heuristic is thus generally smaller than H_{sum}, while still resembling H_{sum}.

The heuristic H_{sum} is a summary of all the subgoal heuristics, resulting in a trade off on progress between all subgoals. However, there are domains on which subgoals are best solved sequentially or concurrently. H_{sum}'s neutral position between these two approaches can be a critical weakness, as we will discuss in the following two subsections.

Often H_{sum} is uninformative when there is negative interaction between subgoals, so that advances on one subgoal lead to matching degradations on another. This problem can be caused, for instance, by competition for non-consumable resources. For one example, on domains where a truck needs to travel to multiple locations, the truck is a resource in the sense that it cannot be present simultaneously at several locations. As the truck moves closer to one subgoal location, it may be moving further from another, resulting in no net effect on H_{sum}. This transportation theme appears in many benchmark domains (Helmert 2003). For another example, on benchmark domains such as *Freecell*, *Airport* and *Sokoban*, free space is another form of resource which cannot be occupied simultaneously by multiple objects. In such cases, H_{sum} can be misleading because it fails to take into account the interaction between competing subgoals—in particular, H_{sum} can fail to encourage seeking the subgoals one at a time to minimize resource competition.

Combination via MIN

H_{count} addresses these issues partially. However, this heuristic alone does not order enough state pairs, as many useful actions do not achieve subgoals immediately. For this reason, tie-breaking by measuring the distance to the nearest subgoal, as in H_{min} is necessary for an effective heuristic focusing on subgoal count. H_{min}-style approaches to conjunctive goals avoid a central problem of H_{sum} by concentrating on a single subgoal at a time. This idea underlies the approach generally used in planners in the early 1970s, as discussed by Sacerdoti (1975).

The recent planner FF-L (Hoffmann, Porteous, & Sebastia 2004) applies an H_{min}-style approach to a set of subgoals found by automatically extracted planning landmarks for the problem—these landmarks include the top-level goal conjuncts. Landmarks are propositions which must be made true at some point by any successful plan. Hoffmann et al. (2004) have developed methods for automatically finding planning landmarks. Unlike the top-level subgoals, a landmark need not be maintained once it is achieved. However, landmarks can be treated like conjunctive subgoals with a small problem modification as we discussed in the introduction. Consequently, our techniques can be directly applied to leveraging automatically discovered planning landmarks.

As discussed in more detail below in the empirical results section, FF-L employs a heuristic approach similar to H_{min} to seek a conjunction of landmarks, showing a substantial empirical gain over state-of-the-art planners that do not leverage landmarks across a wide range of benchmarks.

A major weakness of H_{min} is that H_{min} is insensitive to positive and negative effects on subgoals that are not currently closest.

Another form of this weakness lies in the heuristic focusing too absolutely on retaining previously achieved subgoals[5]. This is a critical problem in domains where negative interaction between subgoals is strong, and a sequence of irreversibly achieved subgoals is hard or impossible to find (Korf 1987).

Our simple method of minimizing H_k addresses these weaknesses, while retaining approximately the strength described just above. With values of k less than 1, H_k trades off both behaviors of H_{min} and H_{sum}. When k is small enough, H_k approximates a natural extension to H_{min}:

$$s_1 \preceq_{\text{minsort}} s_2 \quad \text{iff} \quad \text{sort}(\vec{h}(s_1)) \preceq_{\text{lex}} \text{sort}(\vec{h}(s_2)),$$

where the function sort($\vec{v}$) sorts a vector $\vec{v}$ in non-decreasing order, $\vec{h}(s)$ represents the vector $\langle h_1(s), \ldots, h_{|G|}(s)\rangle$, and $\preceq_{\text{lex}}$ represents the lexicographic order.

$\preceq_{\text{minsort}}$ can be approximated by $\preceq_{H_k}$ with k near zero:

$$\forall\, s_1 \preceq_{\text{minsort}} s_2, \exists\, \delta, \forall\, 0 < k < \delta, s_1 \preceq_{H_k} s_2$$

Combination via MAX

On domains such as *Blocksworld* or *24-puzzle* there are negative interactions between subgoals, but unlike the transportation domains, a subgoal far from completion suggests the need to destroy many already achieved subgoals. Work to complete nearby subgoals when such distant subgoals remain distant is often wasted effort. The H_{sum} conjunctive heuristic may wrongly reward progress toward a subgoal that must be later destroyed. The above-mentioned subgoal count measure does not handle this well, either, because it becomes very important which subgoal is achieved next, not just that one is achieved. Work to order the subgoals to avoid this problem (i.e., so that achieved subgoals need not generally be reversed) has met only limited success (Koehler & Hoffmann 2000; Korf 1987).

In such domains, H_{max} may reflect the true distance more accurately. Note, for instance, that greedily reducing the difficulty of the hardest problem optimally solves the famous *Sussman anomaly* in the *Blocksworld* (as described, e.g., by Korf (1987)).

H_{max} represents a strategy to concurrently solve the subgoals, switching focus from a subgoal once enough progress is made that it is not the furthest. The critical need for concurrency occurs on some domains, typically with consumable, non-exclusive resources such as fuel or time. For example, on problems of scheduling interdependent jobs with a deadline and limited facility resources, following the H_{max} heuristic is equivalent to focusing on the *critical path*.

Previously H_{max} has not been explicitly noted for the above advantages, but has been favored mainly due to its admissibility, finding application in optimal planning. For an example use of H_{max}, one setting of the heuristic planner HSP uses H_{max} (Bonet & Geffner 2001)—this setting

[5]The H_{min}-style approach in FF-L avoids this weakness, as discussed below.

was found less successful than H_{sum} on most domains. This is also the type of combination heuristic used implicitly in GRAPHPLAN's backward phase (Blum & Furst 1997; Haslum & Geffner 2000). A third example is the typical use of multiple pattern databases in search (Korf 2000) and planning (Edelkamp 2002), as discussed above.

A significant weakness of $H_{\max}$ is, similarly to $\preceq_{\min}$ again, they generally ignore positive or negative interactions with subgoals that are not as distant as the furthest subgoal.

Our simple method H_k, with values of $k > 1$, addresses these weaknesses, while retaining approximately the strength described just above, by mixing the behaviors of $H_{\max}$ and H_{sum}. H_k with large values of k approximates a natural extension of $H_{\max}$:

$$s_1 \preceq_{\text{maxsort}} s_2 \quad \text{iff} \quad \text{rsort}(\vec{h}(s_1)) \preceq_{\text{lex}} \text{rsort}(\vec{h}(s_2)),$$

where the function rsort($\vec{v}$) sorts a vector $\vec{v}$ in non-increasing order, $\vec{h}(s)$ represents the vector $\langle h_1(s), \ldots, h_{|G|}(s)\rangle$, and $\preceq_{\text{lex}}$ represents the lexicographic order.

$\preceq_{\text{maxsort}}$ makes more distinctions than $\preceq_{\max}$ does. Our results suggest that this difference can be critical. $\preceq_{\text{maxsort}}$ can be approximated by $\preceq_{H_k}$ with large values of k:

$$\forall\ s_1 \preceq_{\text{maxsort}} s_2, \exists\ \delta, \forall\ k > \delta, s_1 \preceq_{H_k} s_2$$

Empirical Results

Planners We evaluate the performance of our conjunctive heuristics and previous methods on the same search algorithm that is implemented in FF (Hoffmann & Nebel 2001). The core of FF's search algorithm is *enforced hill-climbing*, which is incomplete but often very fast[6]. Unlike pure hill-climbing which repeatedly selects single actions with the best one-step-look-ahead heuristic value (which may be worse than the heuristic value of the current state) and often has difficulty with local minima and plateaus, enforced hill-climbing repeatedly uses breadth-first search to find *action sequences* that lead to states with heuristic values that are strictly better than the current state.

We replace the heuristic function of FF in turn with $H_{\min}$, $H_{\max}$, or H_k, and call the resulting planners MIN, MAX, and K, respectively. We compare the performance of these planners with FF, as well as the recent landmark-based planner FF-L (Hoffmann, Porteous, & Sebastia 2004).

FF-L computes and utilizes landmarks, as introduced in the section "Combination via MIN" on page 3. FF-L also computes a partial order $\prec_L$ on landmarks. FF-L maintains a set L of landmarks that have not been achieved at least once, and directs a base planner to achieve the landmarks in L, as follows. FF-L constructs a disjunction of all the unachieved landmarks that are minimal in the ordering $\prec_L$, i.e., the disjunction over $\{l \mid \forall m \in L, m \not\prec_L l\}$. After the base planner achieves this disjunction, the achieved landmarks are removed from L, and the process is repeated until L is empty. In the final stage, FF-L uses the base planner to simultaneously achieve all the original goal conjuncts again, as some may have been reversed since they were achieved. The last step is necessary because FF-L does not force the base planner to retain achieved landmarks.

When the base planner is FF, the heuristic to the disjunction of unachieved landmarks is the minimum of the heuristics for each landmark. FF-L's behaviors is thus similar to MIN, with five distinctions. First, FF-L considers all the landmarks, which is a superset of the subgoals MIN considers. Second, FF-L does not have the strong bias of MIN on retaining achieved subgoals. Third, unlike MIN, FF-L fails to realize that a problem is unsolvable unless every landmark is unreachable (according to the heuristic) from the current state. Thus, FF-L exhibits a stronger form of MIN's critical weakness of insensitivity to effects on subgoals that are not currently closest—FF-L will happily select actions to achieve nearby subgoals while rendering further subgoals unreachable. Fourth, FF-L's heuristic computation is generally quicker than MIN's, because MIN does a separate heuristic computation for every subgoal, each of which takes at least the time for the heuristic computation of the disjunction. Fifth, in FF-L the branching factor for search is typically greatly reduced because FF's heuristic computation has a side effect of recognizing *helpful actions* (Hoffmann & Nebel 2001), so that only a small fraction of all actions are considered during search. In FF-L only the actions useful for the closest subgoal will be considered helpful actions.

FF-L computes and orders a set of landmarks—none of the other planners we evaluate here use such a computation, relying instead on direct heuristic measures of distance to the top-level goal conjuncts. Yet, surprisingly, our planner K consistently improves upon FF-L in success ratio. This suggests that landmarks are not the critical element in achieving the reported performance for FF-L.

Domains We test the mentioned planners on a wide range of benchmark planning domains encoded in PDDL. Our test suite covers all the domains that FF-L was demonstrated on (Hoffmann, Porteous, & Sebastia 2004): *Blocksworld, Blocksworld-no-arm, Depots, Freecell, Grid, Logistics, Rovers, Tireworld.* Please find detailed descriptions of those domains in the referred paper. We also include a representative domain from the search community: *24-puzzle.*

For each of the domains, we generate a random set of problem instances, and present success ratio, average runtime, and average plan length for each of the planners. We set the runtime bound long enough on each domain for the success ratio to level-off for most planners. The number of random instances and runtime limit for each domain can be found in Table 1.

We use the same parameters as (Hoffmann, Porteous, & Sebastia 2004) to generate random instances for the domains demonstrated there. For the other domain, *24-puzzle*, we generate initial states uniformly at random, and assume that half of the generated instances are solvable. If a domain can scale on size, we have space only to present the results on the largest size tested in (Hoffmann, Porteous, & Sebastia 2004). The results of planners are typically less distinguishable on smaller-sized instances.

Heuristic-search planners are generally fast enough when they succeed, and otherwise often either fail quickly or run

[6] In case the enforced hill-climbing fails, which does not happen often, FF resorts to an expensive but complete search.

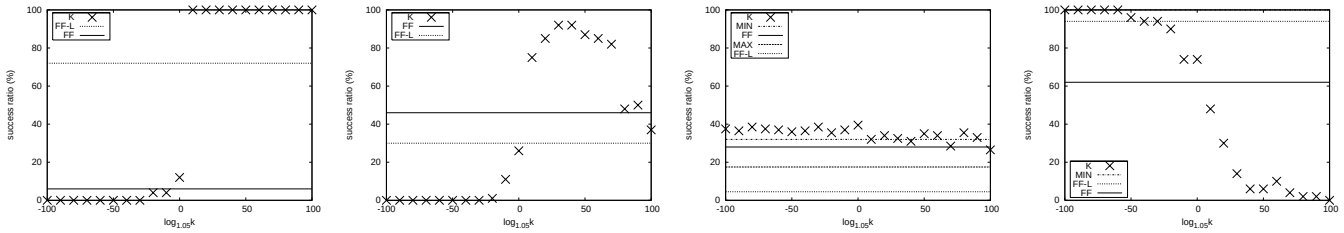

Figure 1: *Success Ratio for Blocksworld, 24-puzzle, Freecell, and Grid.*

	Blocksworld		**24-Puzzle**		**Freecell**		**Grid**	
Sample Size	50		100		200		50	
Time Limit	1000		1000		2000		2000	
Planner	**Runtime (Plan Length)**							
FF-L	1	(94)	115	(384)	6	(126)	94	(241)
MIN	-	-	-	-	611	(151)	237	(198)
k=1.05^{-100}	-	-	-	-	362	(162)	393	(266)
k=1.05^{-50}	-	-	-	-	353	(163)	506	(265)
k=1.05^{-30}	-	-	-	-	311	(164)	679	(268)
k=1.05^{-10}	218	(189)	489	(389)	380	(163)	807	(278)
k=1	93	(156)	302	(361)	357	(156)	812	(276)
FF	235	(129)	139	(281)	453	(133)	1025	(235)
k=1.05^{10}	26	(182)	162	(409)	449	(176)	1079	(305)
k=1.05^{30}	5	(122)	100	(406)	392	(172)	1480	(235)
k=1.05^{50}	7	(139)	91	(426)	521	(180)	1369	(248)
k=1.05^{100}	13	(164)	291	(285)	582	(171)	-	-
MAX	-	-	-	-	1053	(151)	-	-

Table 1: *Number of random instances tested on, runtime limit (in seconds), average runtime (in seconds), and average plan length (enclosed in parentheses). An entry of "-" represents that no instance is solved. The success ratio corresponding to any entry can be found in Figure 1.*

out of memory slowly. We therefore are focused here on increasing the success ratio in difficult domains. The results on success ratio are presented in Figure 1, in which the x axis, representing the value of k used by K, is shown in logarithmic scale. The results on runtime and plan length can be found in Table 1. Runtimes and plan lengths should be interpreted carefully. A planner that appears slower may be solving far more problems, including some that are harder. Only solved problems are included in the planner's average runtime and plan length, giving an advantage to planners that fail on hard problems. We consider plan quality the least important of the three measures; but nonetheless neither FF nor FF-L consistently dominates K in plan length across the choices of k that yield superior success ratio.

Blocksworld, Blocksworld-no-arm, and Depots In the famous *Blocksworld* domain, a robot arm must rearrange towers of blocks. The *Blocksworld-no-arm* domain is an encoding variant of *Blocksworld.* Without explicit reference to a robot arm, FF's heuristic presents a different *local search topology* (Hoffmann 2002). The *Depots* domain enriches *Blocksworld* with transportation, so that blocks can be delivered between any pair of locations by trucks. We present results only on *Blocksworld,* as the results on the other domains are very similar.

Each random problem instance consists of 30 blocks. As shown in the left part of Figure 1, $H_{\max}$-style heuristics work much better than H_{sum} and $H_{\min}$-style heuristics on the *Blocksworld* domain. FF only solves 6% of the problems. When k equals one, K performs slightly better than FF. As k increases from one, the success ratio jumps up to 100%. With the help of landmarks, FF-L gets a significant higher success ratio (72%) than FF, but is still significantly less reliable than H_k with $k > 1$, even though no landmarks are leveraged by the latter. Neither MAX nor MIN solves any instance. This suggests $H_{\max}$ itself does not order enough pairs to be useful.

Twenty-four Puzzle The SLIDING-TILE puzzles have long been testbeds for domain-specific search. The *24-puzzle*, which contains almost 10^{25} states, was first optimally solved by Korf (1996) with human-designed domain-specific heuristics. Here we will show that K with suitable values of $k > 1$, employing FF's domain-independent heuristic on the goal conjuncts, can solve most randomly generated instances sub-optimally.

We generate 200 random instances and assume half of them are solvable. The success ratio results shown in the second part of Figure 1 indicate that a suitable trade-off between $H_{\max}$ and H_{sum} best solves *24-puzzle.* This is reasonable since often correctly positioned tiles must be temporarily moved out of place to achieve other subgoals. 92% of the solvable instances were solved by K with $k = 1.05^{30}$ (likewise for $k = 1.05^{40}$). The performance of K drops sharply for larger or smaller k. FF, at 46% success, outperforms H_{sum} (K with $k = 1$) at 26%, suggesting that FF is benefiting from considering positive interactions. FF-L does not find useful landmarks other than the top-level subgoals, achieving success ratio 24%. Neither MIN nor MAX solves any instance. The performance difference between FF-L and MIN is likely due to FF-L's tolerance for destroying achieved subgoals, as discussed above.

Freecell The *Freecell* domain is an adaptation of the SOLITAIRE card game. Stacks of cards need to be rearranged, much like in *Blocksworld.* But certain rules must be respected when moving the cards, one of which introduces a non-consumable resource: Only a limited set of *free cells* can be temporarily occupied by single cards, unlike *Blocksworld,* where any number of blocks can be put on the table. In this sense, this domain does not yield to either sequentially solving subgoals or to solving them entirely concurrently (due to congestion).

In each of our test instances, there are four suits and each suit has 16 cards. They initially form up to ten stacks, and need to be rearranged into four goal stacks, with help of five free cells. As shown in the third part in Figure 1, the success

ratio is 32% for MIN, 28% for FF, 17.5% for MAX, and 4.5% for FF-L. Neither a sequential (MIN) nor a concurrent (MAX) strategy works particularly well on the *Freecell* domain. The trade-off method K does not find significant improvement either. The success ratio of K is slightly higher than that of MIN with $k \leq 1$, and is around that of MIN with $k > 1$.

Grid In the *Grid* domain a robot needs to transport keys on a two-dimensional grid. Additionally, some grid positions must be opened by matching keys to allow the robot to go through. Each test instance is randomly generated with 6 keys on a 10×10 grid. The right part in Figure 1 shows that both MIN and K with small values of k solve all problem instances. FF-L appears less reliable, solving 94%. For larger k, the success ratio of K drops dramatically. The success ratio of FF (62%) is slightly worse than K's 74% when $k = 0$. These results clearly demonstrate that the *Grid* domain is best solved by sequentially attacking the subgoals.

Logistics, Rovers, and Tireworld Unlike the domains presented above, the *Logistics*, *Rovers* and *Tireworld* domains are solved with 100% success ratio by FF, K or FF-L given enough time. Since our focus is the success ratio, we will only summarize our findings on these domains.

On *Logistics* and *Rovers*, K with small values of k have minor improvements over FF upon speed. The performance of K drops dramatically as k goes up from one. On *Tireworld*, K is slightly slower than FF with $k = 0$, and gets worse when k is greater or smaller. On all three domains, FF-L is much quicker than FF or K.

Both the domains *Logistics* and *Rovers* share the transportation theme with *Grid*. But unlike on *Grid*, where two locations can be arbitrarily far away, in these domains each subgoal can be achieved in a small number of steps. This also holds for *Tireworld*[7]. In consequence, enforced hill-climbing solves a problem on these domains by a sequence of searches, each bound by a small constant in depth. The factors limiting scaling performance are then the heuristic computation for each state and the branching factor in search. As discussed in the previous subsection, "Planners", FF-L has an advantage on these two factors.

Conclusion

Subgoal interactions have been widely studied in planning, e.g. in (Korf 1987; Cheng & Irani 1989; Kambhampati, Ihrig, & Srivastava 1996; Blum & Furst 1997; Koehler & Hoffmann 2000). Here, we first explicitly formulate this difficult problem as building conjunctive heuristics. We showed that a straightforward trade-off method adapts to a wide range of domains with different characteristics of subgoal interactions. Applied to top-level subgoals, our method significantly outperforms two state-of-the-art planners, one of which employs much more complex techniques than ours.

It is apparent from Figure 1 that a simple search over values of the parameter k will easily identify, for each domain considered here, a value that leads to state-of-the-art planning performance.

Our method can be potentially applied beyond the scope of classical planning, as heuristics can measure not only the number of plan steps, but also other forms of cost such as time or resources (as explored by, e.g., (Hoffmann 2003)), as well as *expected cost* (Bonet & Geffner 2003) in probabilistic planning.

[7] On *Tireworld* a number of flat tires must be replaced, each requiring a fixed procedure consisting of several steps.

References

Blum, A., and Furst, M. L. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90(1-2):281–300.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1-2):5–33.

Bonet, B., and Geffner, H. 2003. Faster heuristic search algorithms for planning with uncertainty and full feedback. In *IJCAI*, 1233–1238.

Cheng, J., and Irani, K. B. 1989. Ordering problem subgoals. In *IJCAI*, 931–936.

Culberson, J. C., and Schaeffer, J. 1996. Searching with pattern databases. In *Canadian Conference on AI*, 402–416.

Edelkamp, S. 2002. Symbolic pattern databases in heuristic search planning. In *AIPS*, 274–283.

Gerevini, A.; Saetti, A.; and Serina, I. 2003. Planning through stochastic local search and temporal action graphs in LPG. *JAIR* 20:239–290.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *AIPS*, 140–149.

Helmert, M. 2003. Complexity results for standard benchmark domains in planning. *Artificial Intelligence* 143(2):219–262.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *ICAPS*, 161–170.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR* 14:253–302.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *JAIR* 22:215–278.

Hoffmann, J. 2002. Local search topology in planning benchmarks: A theoretical analysis. In *AIPS*, 92–100.

Hoffmann, J. 2003. The metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *JAIR* 20:291–341.

Kambhampati, S.; Ihrig, L. H.; and Srivastava, B. 1996. A candidate set based analysis of subgoal interactions in conjunctive goal planning. In *AIPS*, 125–133.

Koehler, J., and Hoffmann, J. 2000. On reasonable and forced goal orderings and their use in an agenda-driven planning algorithm. *JAIR* 12:338–386.

Korf, R. E., and Taylor, L. A. 1996. Finding optimal solutions to the twenty-four puzzle. In *AAAI*, 1202–1207.

Korf, R. E. 1987. Planning as search: A quantitative approach. *Artificial Intelligence* 33(1):65–88.

Korf, R. E. 2000. Recent progress in the design and analysis of admissible heuristic functions. In *AAAI*, 1165–1170.

McAllester, D. A., and Rosenblitt, D. 1991. Systematic nonlinear planning. In *AAAI*, 634–639.

Nguyen, X., and Kambhampati, S. 2001. Reviving partial order planning. In *IJCAI*, 459–466.

Sacerdoti, E. D. 1975. The nonlinear nature of plans. In *IJCAI*, 206–214.