A Theoretical Framework for Studying Random Walk Planning

Hootan Nakhost
University of Alberta, Edmonton, Canada
nakhost@ualberta.ca

Martin Müller
University of Alberta, Edmonton, Canada
mmueller@ualberta.ca

Abstract

Random walks are a relatively new component used in several state of the art satisficing planners. Empirical results have been mixed: while the approach clearly outperforms more systematic search methods such as weighted A* on many planning domains, it fails in many others. So far, the explanations for these empirical results have been somewhat ad hoc. This paper proposes a formal framework for comparing the performance of random walk and systematic search methods. Fair homogenous graphs are proposed as a graph class that represents characteristics of the state space of prototypical planning domains, and is simple enough to allow a theoretical analysis of the performance of both random walk and systematic search algorithms. This gives well-founded insights into the relative strength and weaknesses of these approaches. The close relation of the models to some well-known planning domains is shown through simplified but semi-realistic planning domains that fulfill the constraints of the models.

One main result is that in contrast to systematic search methods, for which the branching factor plays a decisive role, the performance of random walk methods is determined to a large degree by the Regress Factor, the ratio between the probabilities of progressing towards and regressing away from a goal with an action. The performance of random walk and systematic search methods can be compared by considering both branching and regress factors of a state space.

Random Walks in Planning

Random walks, which are paths through a search space that follow successive randomized state transitions, are a main building block of prominent search algorithms such as Stochastic Local Search techniques for SAT (Selman, Levesque, and Mitchell 1992; Pham et al. 2008) and Monte Carlo Tree Search in game playing and puzzle solving (Gelly and Silver 2008; Finnsson and Björnsson 2008; Cazenave 2009).

Inspired by these methods, several recent satisficing planners also utilize random walk (RW) techniques. Identidem (Coles, Fox, and Smith 2007) performs a hill climbing search that uses random walks to escape from plateaus or saddle points. All visited states are evaluated using a heuristic function. Random walks are biased towards states with lower heuristic value. Roamer (Lu et al. 2011) enhances its best-first search (BFS) with random walks, aiming to escape from *search plateaus* where the heuristic is uninformative.

Arvand (Nakhost and Müller 2009) takes a more radical approach: it relies exclusively on a set of random walks to determine the next state in its local search. For efficiency, it only evaluates the endpoints of those random walks. Arvand also learns to bias its random walks towards more promising actions over time, by using the techniques of *Monte Carlo Deadlock Avoidance* (MDA) and *Monte Carlo with Helpful Actions* (MHA). In (Nakhost, Hoffmann, and Müller 2012), the local search of Arvand2 is enhanced by the technique of *Smart Restarts*, and applied to solving Resource Constrained Planning (RCP) problems. The hybrid *Arvand-LS* system (Xie, Nakhost, and Müller 2012) combines random walks with a local greedy best first search.

Compared to all other tested planners, Arvand2 performs much better in RCP problems (Nakhost, Hoffmann, and Müller 2012), which test the ability of planners in utilizing scarce resources. In IPC domains, RW-based planners tend to excel on domains with many paths to the goal. For example, scaling studies in (Xie, Nakhost, and Müller 2012) show that RW planners can solve much larger problem instances than other state of the art planners in the domains of *Transport*, *Elevators*, *Openstacks*, and *Visitall*. However, the planners perform poorly in *Sokoban*, *Parking*, and *Barman*, puzzles with a small solution density in the search space.

While the success of RW methods in related research areas such as SAT and Monte Carlo Tree Search serves as a good general motivation for trying them in planning, it does not provide an explanation for why RW planners perform well. Previous work has highlighted three main advantages of random walks for planning:

- Random walks are more effective than systematic search approaches for escaping from regions where heuristics provide no guidance (Coles, Fox, and Smith 2007; Nakhost and Müller 2009; Lu et al. 2011).
- Increased sampling of the search space by random walks adds a beneficial *exploration* component to balance the *exploitation* of the heuristic in planners (Nakhost and Müller 2009).
- Combined with proper *restarting* mechanisms, random walks can avoid most of the time wasted by systematic

search in dead ends. Through restarts, random walks can rapidly back out of unpromising search regions (Coles, Fox, and Smith 2007; Nakhost, Hoffmann, and Müller 2012).

These explanations are intuitively appealing, and give a qualitative explanation for the observed behavior on planning benchmarks such as IPC and IPC-2011-LARGE (Xie, Nakhost, and Müller 2012). Typically, random walk planners are evaluated by measuring their coverage, runtime, or plan quality in such benchmarks.

Studying Random Walk Methods

There are many feasible approaches for gaining a deeper understanding of these methods.

- Scaling studies, as in Xie et al. (2012).
- Algorithms combining RW with other search methods, as in (Lu et al. 2011; Valenzano et al. 2011).
- Experiments on small finite instances where it is possible to "measure everything" and compare the choices made by different search algorithms.
- Direct measurements of the benefits of RW, such as faster escape from plateaus of the heuristic.
- A theoretical analysis of how RW and other search algorithms behave on idealized classes of planning problems which are amenable to such analysis.

The current paper pursues the latter approach. The main goal is a careful theoretical investigation of the first advantage claimed above - the question of how RW manage to escape from plateaus faster than other planning algorithms.

A First Motivating Example

As an example, consider the following well-known plateau for the FF heuristic, h_{FF}, discussed in (Helmert 2004). Recall that h_{FF} estimates the goal distance by solving a relaxed planning problem in which all the negative effects of actions are ignored. Consider a transportation domain in which trucks are used to move packages between n locations connected in a single chain $c_1, \cdots, c_n$. The goal is to move one package from c_n to c_1. Figure 1 shows the results of a basic scaling experiment on this domain with $n = 10$ locations, varying the number of trucks T from 1 to 20. All trucks start at c_2. The results compare basic Monte Carlo Random Walks (MRW) from Arvand-2011 and basic Greedy Best First Search (GBFS) from LAMA-2011. Figure 1 shows how the runtime of GBFS grows quickly with the number of trucks T until it exceeds the memory limit of 64 GB. This is expected since the effective branching factor grows with T. However, the increasing branching factor has only little effect on MRW: the runtime grows only linearly with T.

Choice of Basic Search Algorithms

All the examples in this paper use state of the art implementations of basic, unenhanced search methods. GBFS as implemented in LAMA-2011 represents systematic search methods, and the MRW implementation of Arvand-2011 represents random walk methods. Both programs use h_{FF} for their evaluation. All other enhancements such as preferred operators in LAMA and Arvand, multi-heuristic search in LAMA, and MHA in Arvand are switched off.

The reasons for selecting this setup are: 1. A focus on theoretical models that can explain the substantially different behavior of random walk and systematic search methods. Using simple search methods allows a close alignment of experiments with theoretical results. 2. Enhancements may benefit both methods in different ways, or be only applicable to one method, so may confuse the picture. 3. A main goal here is to understand the behavior of these two search paradigms in regions where there is a lack of guiding information, such as plateaus. Therefore, in some examples even a blind heuristic is used. While enhancements can certainly have a great influence on search parameters such as branching factor, regress factor, and search depth, the fundamental differences in search behavior will likely persist across such variations.

Contributions of this Paper

Regress factor and goal distance for random walks: The key property introduced to analyze random walks is the *regress factor* rf, the ratio of two probabilities: *progressing* towards a goal and *regressing* away from it. Besides rf, the other key variable affecting the average runtime of basic random walks on a graph is the *largest goal distance* D in the whole graph, which appears in the exponent of the expected runtime.

Homogenous graph model: In the *homogenous graph* model, the regress factor of a node depends only on its goal distance. Theorem 3 shows that the runtime of RW mainly depends on rf. As an example, the state space of Gripper is close to a homogenous graph.

Bounds for other graphs: Theorem 4 extends the theory to compute upper bounds on the hitting time for graphs which are not homogeneous, but for which bounds on the progress and regress chances are known.

Strongly homogenous graph model: In *strongly homogenous graphs*, almost all nodes share the same rf. Theorem 5 explains how *rf* and D affect the hitting time. A transport example is used for illustration.

Model for Restarting Random Walks: For large values of D, *restarting random walks* (RRW) can offer a substantial performance advantage. At each search step, with probability r a RRW restarts from a fixed initial state s. Theorem 6 proves that the expected runtime of RRW depends only on the goal distance of s, not on D.

Background and Notation

Notation follows standard references such as (Norris 1998). Throughout the paper the notation $P(e)$ denotes the probability of an event e occuring, $G = (V, E)$ is a directed graph, and $u, v \in V$ are vertices.

Definition 1 (Markov Chain). *The discrete-time random process* $X_0, \ldots, X_N$ *defined over a set of states* S *is* $Markov(S, \mathbb{P})$ *iff* $P(X_n = j_n | X_{n-1} = j_{n-1}, \ldots, X_0 = j_0) = P(X_n = j_n | X_{n-1} = j_{n-1})$. *The matrix* $\mathbb{P}(p_{ij})$ *where* $p_{ij} =$

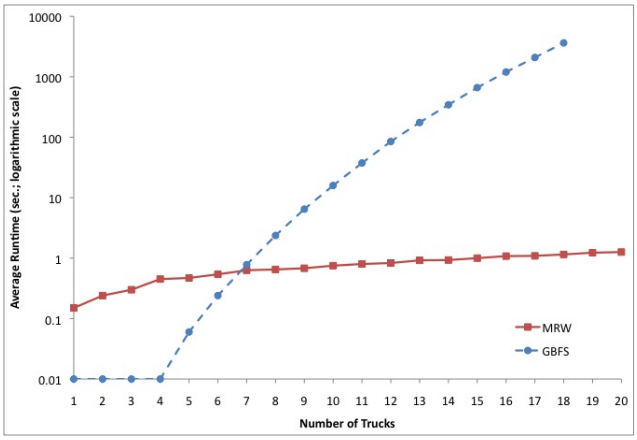

Figure 1: Average runtime of GBFS and MRW varying the number of trucks (x-axis) in Transport domain. Missing data means memory limit exceeded.

$P(X_n = j_n | X_{n-1} = i_{n-1})$ are the transition probabilities of the chain. In time-homogenous Markov chains as used in this paper, $\mathbb{P}$ does not depend on n.

Definition 2 (Distance d_G). *$d_G(u,v)$ is the length of a shortest path from u to v in G. The distance $d_G(v)$ of a single vertex v is the length of a longest shortest path from a node in G to v: $d_G(v) = max_{x \in V} d_G(x,v)$.*

Definition 3 (Successors). *The successors of $u \in V$ is the set of all vertices in distance 1 of u:*
$S_G(u) = \{v | v \in V \wedge d_G(u,v) = 1\}$.

Definition 4 (Random Walk). *A random walk on G is a Markov chain $Markov(V, \mathbb{P})$ where $p_{uv} = \frac{1}{|S_G(u)|}$ if $(u,v) \in E$, and $p_{uv} = 0$ if $(u,v) \notin E$.*

The *restarting random walk* model used here is a random walk which *restarts* from a fixed initial state s with probability r at each step, and uniformly randomly chooses among neighbour states with probability $1 - r$.

Definition 5 (Restarting Random Walk). *Let $s \in V$ be the initial state, and $r \in [0,1]$. A restarting random walk $RRW(G,s,r)$ is a Markov chain M_G with states V and transition probabilities p_{uv}:*

$$p_{uv} = \begin{cases} \frac{1-r}{|S_G(u)|} & \text{if } (u,v) \in E, v \neq s \\ r + \frac{1-r}{|S_G(u)|} & \text{if } (u,v) \in E, v = s \\ 0 & \text{if } (u,v) \notin E, v \neq s \\ r & \text{if } (u,v) \notin E, v = s \end{cases}$$

A RW is the special case of RRW with $r = 0$.

Definition 6 (Hitting Time). *Let $M = X_0, X_1, \ldots, X_N$ be $Markov(S, \mathbb{P})$, and $u, v \in S$. Let $H_{uv} = min\{t \geq 1 : X_t = v \wedge X_0 = u\}$. Then the hitting time h_{uv} is the expected number of steps in a random walk on G starting from u which reaches v for the first time: $h_{uv} = E[H_{uv}]$.*

Definition 7 (Unit Progress Time). *The unit progress time u_{uv} is the expected number of steps in a random walk after reaching u for the first time until it first gets closer to v. Let $R = RRW(G,s,r)$. Let $U_{uv} = min\{t \geq H_{su} : d_G(X_t, v) = d_G(u,v) - 1\}$. Then $u_{uv} = E[U_{uv}]$.*

Definition 8 (Progress, Regress and Stalling Chance; Regress Factor). *Let $X : V \to V$ be a random variable with the following probability mass function:*

$$P(X(u) = v) = \begin{cases} \frac{1}{|S_G(u)|} & \text{if } (u,v) \in E \\ 0 & \text{if } (u,v) \notin E \end{cases} \quad (1)$$

Let X_u be short for $X(u)$. The progress chance $pc(u,v)$, regress chance $rc(u,v)$, and stalling chance $sc(u,v)$ of u regarding v, are respectively: the probabilities of getting closer, further away, or staying at the same distance to v after one random step at u.

$$pc(u,v) = P(d_G(X_u, v) = d_G(u,v) - 1)$$
$$rc(u,v) = P(d_G(X_u, v) = d_G(u,v) + 1)$$
$$sc(u,v) = P(d_G(X_u, v) = d_G(u,v))$$

In a Markov Chain, the probability transitions play a key role in determining the hitting time. In all the models considered here, the movement in the chain corresponds to moving between different goal distances. Therefore it is natural to choose progress and regress chances as the main properties. The regress factor of u regarding v is $rf(u,v) = \frac{rc(u,v)}{pc(u,v)}$ if $pc(u,v) \neq 0$, and undefined otherwise.

Theorem 1. *(Norris 1998) Let M be $Markov(V, \mathbb{P})$. Then for all $u, v \in V$, $h_{uv} = 1 + \sum_{x \in V} p_{ux} h_{xv}$.*

Theorem 2. *Let $s \in V$, $D = d_G(u,v)$, $R = RRW(G,s,r)$, $V_d = \{x : x \in V \wedge d_G(x,v) = d\}$, and $P_d(x)$ be the probability of x being the first node in V_d reached by R. Then the hitting time $h_{uv} = \sum_{d=1}^{D} \sum_{x \in V_d} P_d(x) u_{xv}$.*

Proof. Let H_{uv} and X_d be two random variables respectively denoting the length of a RRW that starts from u and ends in v for the first time, and the first vertex $x \in V_d$ reached by R. Then

$$H_{uv} = \sum_{d=1}^{D} \sum_{x \in V_d} 1_{\{X_d\}}(x) U_{xv} \quad (2)$$

where U_{xv} is a random variable measuring the length of the fragment of the walk starting from x and ending in a smaller goal distance for the first time, and $1_{\{X_d\}}(x)$ is an indicator random variable which returns 1 if $X_d = x$ and 0 if $X_d \neq x$. Since random variables x and U_{xv} are independent,

$$E[H_{uv}] = \sum_{d=1}^{D} \sum_{x \in V_d} E[1_{\{X_d\}}(x)] E[U_{xv}]$$

$$h_{uv} = \sum_{d=1}^{D} \sum_{x \in V_d} P_d(x) u_{xv}$$

Heuristic Functions, Plateaus, Exit Points and Exit Time

What is the connection between the models introduced here and plateaus in planning? Using the notation of (Hoos and Stützle 2004), let the heuristic value $h(u)$ of vertex u be the estimated length of a shortest path from u to a goal vertex v. A *plateau* $P \subseteq V$ is a connected subset of states which share the same heuristic value h_P. A state s is an *exit point* of P if $s \in S_G(p)$ for some $p \in P$, and $h(s) < h_P$. The *exit time* of a random walk on a plateau P is the expected number of steps in the random walk until it first reaches an exit point. The problem of finding an exit point in a plateau is equivalent to the problem of finding a goal in the graph consisting of P plus all its exit points, where the exit points are goal states. The expected exit time from the plateau equals the hitting time of this problem.

Fair Homogenous Graphs

A fair homogeneous (FH) graph G is the main state space model introduced here. *Homogenuity* means that both progress and regress chances are constant for all nodes at the same goal distance. *Fairness* means that an action can change the goal distance by at most one.

Definition 9 (Homogenous Graph). *For $v \in V$, G is v-homogeneous iff there exist two real functions $pc_G(x,d)$ and $rc_G(x,d)$, mapping $V \times \{0,1,\ldots,d_G(v)\}$ to the range $[0,1]$, such that for any two vertices $u,x \in V$ with $d_G(u,v) = d_G(x,v)$ the following two conditions hold:*

1. *If $d_G(u,v) \neq 0$, then $pc_G(u,v) = pc_G(x,v) = pc_G(v,d_G(u,v))$.*
2. *$rc_G(u,v) = rc_G(x,v) = rc_G(v,d_G(u,v))$.*

G is homogeneous iff it is v-homogeneous for all $v \in V$. $pc_G(x,d)$ and $rc_G(x,d)$ are called progress chance and regress chance of G regarding x. The regress factor of G regarding x is defined by $rf_G(x,d) = rc_G(x,d)/pc_G(x,d)$.

Definition 10 (Fair Graph). *G is fair for $v \in V$ iff for all $u \in V$, $pc(u,v) + rc(u,v) + sc(u,v) = 1$. G is fair if it is fair for all $v \in V$.*

Lemma 1. *Let $G = (V,E)$ be FH and $v \in V$. Then for all $x \in V$, h_{xv} depends only on the goal distance $d = d_G(x,v)$, not on the specific choice of x, so $h_{xv} = h_d$.*

Proof. This lemma holds for both RW and RRW. The proof for RRW is omitted for lack of space. Let $p_d = pc_G(v,d)$, $q_d = rc_G(v,d)$, $c_d = sc_G(v,d)$, $D = d_G(v)$, and $V_d = \{x : x \in V \wedge d_G(x,v) = d\}$. The first of two proof steps shows that for all $x \in V_d$, $u_{xv} = u_d$.

Let $I_x(d)$ be the number of times a random walk starting from $x \in V_d$ visits a state with goal distance d before first reaching the goal distance $d-1$, and let $J_x(d)$ be the number of steps between two consecutive such visits. Then, $u_{xv} = E[I_x(d) \times J_x(d) + 1]$. Claim: both $I_x(d)$ and $J_x(d)$ are independent of the specific choice of $x \in V_d$, so $I_x(d) = I(d)$ and $J_x(d) = J(d)$. This implies $u_{xv} = E[I_x(d) \times J_x(d) + 1] = E[I(d) \times J(d) + 1]$ independent of the choice of x, so $u_{xv} = u_d$.

First, the progress chance for all $x \in V_d$ is p_d, therefore $E[I_x(d)] = \frac{1}{p_d} = I(d)$, the expected value of a geometric distribution with the success probability p_d.

Second, $E[J_x(d)] = J(d)$ and therefore $u_{xv} = u_d$ are shown by downward induction for $d = D, \cdots, 1$. For the base case $d = D$, since the random walk can only stall between visits, $E[J_x(D)] = J(D) = 1$. Now assume the claims about J and u hold for $d+1$, so for all $x' \in V_{d+1}$, $E[J_{x'}(d+1)] = J(d+1)$ and $u_{x'} = u_{d+1}$. Call the last step at distance d, before progressing to $d-1$, a *successful d-visit*, and all previous visits, which do not immediately proceed to $d-1$, *unsuccessful d-visits*. After an unsuccessful d-visit, a random walk starting at any $x \in V_d$ stalls at distance d with probability c_d, and transitions to a node with distance $d+1$ with probability q_d, after which it reaches distance d again after an expected u_{d+1} steps. Therefore,

$$E[J_x(d)] = \frac{(c_d + q_d(u_{d+1}+1))}{1-p_d} = J(d)$$

independent of x. As the second proof step, the lemma now follows from Theorem 2:

$$h_{xv} = \sum_{d=1}^{d_G(x,v)} \sum_{k \in V_d} P_d(k) u_{kv} = \sum_{d=1}^{d_G(x,v)} u_d = h_d \tag{3}$$

Theorem 3. *Let $G = (V,E)$ be FH, $v \in V$, $p_i = pc_G(v,i)$, $q_i = rc_G(v,i)$, and $d_G(v) = D$. Then for all $x \in V$,*

$$h_{xv} = \sum_{d=1}^{d_G(x,v)} \left(\beta_D \prod_{i=d}^{D-1} \lambda_i + \sum_{j=d}^{D-1} \left(\beta_j \prod_{i=d}^{j-1} \lambda_i \right) \right)$$

where for all $1 \leq d \leq D$, $\lambda_d = \frac{q_d}{p_d}$, and $\beta_d = \frac{1}{p_d}$.

Proof. According to Lemma 1 and Theorem 1,

$$\begin{aligned} h_0 &= 0 \\ h_d &= p_d h_{d-1} + q_d h_{d+1} + c_d h_d + 1 \quad (0 < d < D) \\ h_D &= p_D h_{D-1} + (1-p_D) h_D + 1 \end{aligned}$$

Let $u_d = h_d - h_{d-1}$, then

$$\begin{aligned} u_d &= \lambda_d u_{d+1} + \beta_d \quad (0 < d < D) \\ u_D &= \beta_D \end{aligned}$$

By induction on d, for $d < D$

$$u_d = \beta_D \prod_{i=d}^{D-1} \lambda_i + \sum_{j=d}^{D-1} \left(\beta_j \prod_{i=d}^{j-1} \lambda_i \right) \tag{4}$$

This is trivial for $d = D-1$. Assume that Equation 4 holds

Robot	Gripper	pc	rc	rf	b
A	full	$\frac{1}{2}$	$\frac{1}{2}$	1	1
A	empty	$\frac{\lvert A\rvert}{\lvert A\rvert+1}$	$\frac{1}{\lvert A\rvert+1}$	$\frac{\mathbf{1}}{\lvert \boldsymbol{A}\rvert}$	$\lvert \boldsymbol{A}\rvert$
B	full	$\frac{1}{2}$	$\frac{1}{2}$	1	1
B	empty	$\frac{1}{\lvert B\rvert+1}$	$\frac{\lvert B\rvert}{\lvert B\rvert+1}$	$\lvert B\rvert$	$\lvert B\rvert$

Table 1: Random walks in One-handed Gripper. $|A|$ and $|B|$ denote the number of balls in A and B.

for $d+1$. Then by Equation 3 for h_{xv},

$$\begin{aligned}
u_d &= \lambda_d \left(\beta_D \prod_{i=d+1}^{D-1} \lambda_i + \sum_{j=d+1}^{D-1} \left(\beta_j \prod_{i=d+1}^{j-1} \lambda_i \right) \right) + \beta_d \\
&= \beta_D \prod_{i=d}^{D-1} \lambda_i + \lambda_d \sum_{j=d+1}^{D-1} \left(\beta_j \prod_{i=d+1}^{j-1} \lambda_i \right) + \beta_d \\
&= \beta_D \prod_{i=d}^{D-1} \lambda_i + \sum_{j=d+1}^{D-1} \left(\beta_j \prod_{i=d}^{j-1} \lambda_i \right) + \beta_d \prod_{i=d}^{d-1} \lambda_i \\
&= \beta_D \prod_{i=d}^{D-1} \lambda_i + \sum_{j=d}^{D-1} \left(\beta_j \prod_{i=d}^{j-1} \lambda_i \right) \\
h_{xv} &= \sum_{d=1}^{d_G(x,v)} \left(\beta_D \prod_{i=d}^{D-1} \lambda_i + \sum_{j=d}^{D-1} \left(\beta_j \prod_{i=d}^{j-1} \lambda_i \right) \right)
\end{aligned}$$

The largest goal distance D and the regress factors $\lambda_i = q_i/p_i$ are the main determining factors for the expected runtime of random walks in homogenous graphs.

Example domain: One-handed Gripper

Consider a one-handed gripper domain, where a robot must move n balls from room A to B by using the actions of picking up a ball, dropping its single ball, or moving to the other room. The highly symmetrical search space is FH. The goal distance determines the distribution of balls in the rooms as well as robot location and gripper status as shown in Table 1. The graph is fair since no action changes the goal distance by more than one. The expected hitting time is given by Theorem 3.

Figure 2 plots the predictions of Theorem 3 together with the results of a scaling experiment, varying n for both random walks and greedy best first search. To simulate the behaviour of both algorithms in plateaus with a lack of heuristic guidance, a blind heuristic is used which returns 0 for the goal and 1 otherwise. Search stops at a state with a heuristic value lower than that of the initial state. Because of the blind heuristic, the only such state is the goal state. The prediction matches the experimental results extremely well. Random walks outperform greedy best first search. The regress factor rf never exceeds b, and is significantly smaller in states with the robot at A and an empty gripper - almost one quarter of all states.

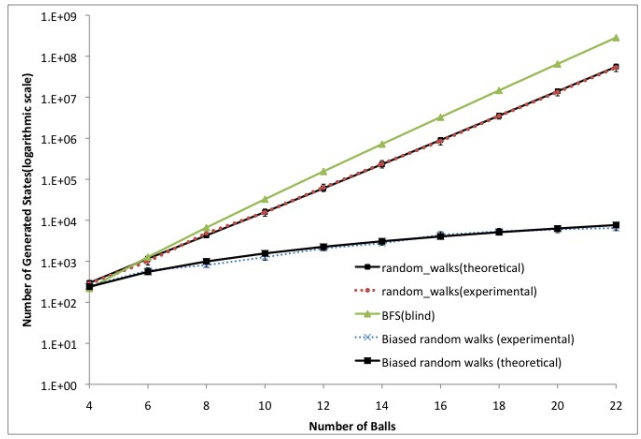

Figure 2: The average number of generated states varying the number of balls (x-axis) in Gripper domain.

Biased Action Selection for Random Walks

Regress factors can be changed by biasing the action selection in the random walk. It seems natural to first select an action type uniformly randomly, then ground the chosen action. In gripper, this means choosing among the balls in the same room in case of the pick up action.

With this biased selection, the search space becomes fair homogenous with $q = p = \frac{1}{2}$. The experimental results and theoretical prediction for such walks are included in Figure 2. The hitting time grows only linearly with n. It is interesting that this natural way of biasing random walks is able to exploit the symmetry inherent in the gripper domain.

Extension to Bounds for Other Graphs

While many planning problems cannot be exactly modelled as FH graphs, these models can still be used to obtain upper bounds on the hitting time in any fair graph G which models a plateau. Consider a corresponding FH graph G' with progress and regress chances at each goal distance d respectively set to the minimum and maximum progress and regress chances over all nodes at goal distance d in G. Then the hitting times for G' will be an upper bound for the hitting times in G. In G', progressing towards the goal is at most as probable as in G.

Theorem 4. *Let $G = (V, E)$ be a directed graph, $s, v \in V$, $R = RRW(G, s, r)$, and $D = d_G(v)$. Let $p_{min}(d)$ and $q_{max}(d)$ be the minimum progress and maximum regress chance among all nodes at distance d of v. Let $G' = (V', E')$ be an FH graph, $v', s' \in V'$, $d_{G'}(v') = D$, $R' = RRW(G', s', r)$, $pc_{G'}(v', d) = p_{min}(d)$, $rc_{G'}(d) = q_{max}(d)$, and $sc_{G'}(d) = 1 - p_{min}(d) - q_{max}(d)$. Then the hitting time of R', $h_{s'v'}$, is a lower bound for the hitting time of R, h_{sv}, i.e., $h_{sv} \leq h'_{s'v'}$ if $d_G(s, v) = d_{G'}(s', v')$.*

Proof. Again, for space reasons only the case $r = 0$ is shown. Let $V_d = \{x | x \in V \wedge d_G(x, v) = d\}$, and assume for all $x \in V_d$, $u_{xv} \leq u'_d$ where u'_d is the unit progress time at

distance d of v'. According to Theorem 2,

$$h_{sv} = \sum_{d=1}^{d_G(s,v)} \sum_{k \in V_d} P_d(x) u_{kv} \leq \sum_{d=1}^{d_{G'}(s',v')} u'_d \leq h'_d$$

To prove $u_{xv} \leq u'_d$ by induction, assume for all $x' \in V_{d+1}$, $u_{x'v} \leq u'_{d+1}$. Then $u_{xv} \leq q_x(u_{d+1}+u_{Iv})+(1-p_x-q_x)u_{Jv}+1$, where I and J are random variables defined over V_d, and p_x and q_x denote the progress and regress chances of x. Let $m = \arg\max_{i \in V_d}(u_{iv})$. Then,

$$\begin{aligned} u_{mv} &\leq q_m(u'_{d+1} + u_{mv}) + (1 - p_m - q_m)u_{mv} + 1 \\ u_{mv} &\leq \frac{q_m}{p_m} u'_{d+1} + \frac{1}{p_m} \leq \frac{q_{max}(d)}{p_{min}(d)} u_{d+1} + \frac{1}{p_{min}(d)} \leq u'_d \end{aligned}$$

Analogously, for the base case $d = D$, for all $x \in V_D$

$$u_{mv} \leq \frac{1}{p_m} \leq \frac{1}{p_{min}(d)} \leq u'_d$$

Fair Strongly Homogeneous Graphs

A fair strongly homogenous (FSH) graph G is a FH graph in which pc and rc are constant for all nodes. FSH graphs are simpler to study and suffice to explain the main properties of FH graphs. Therefore, this model is used to discuss key issues such as dependency of the hitting time on largest goal distance D and the regress factors.

Definition 11 (Strongly Homogeneous Graph). *Given $v \in V$, G is strongly v-homogeneous iff there exist two real functions $pc_G(x)$ and $rc_G(x)$ with domain V and range $[0,1]$ such that for any vertex $u \in V$ the following two conditions hold:*

1. *If $u \neq v$ then $pc(u,v) = pc_G(v)$.*
2. *If $d(u,v) < d_G(v)$ then $rc(u,v) = rc_G(v)$.*

G is strongly homogeneous iff it is strongly v-homogeneous for all $v \in V$. The functions $pc_G(x)$ and $rc_G(x)$ are respectively called the progress and the regress chance of G regarding x. The regress factor of G regarding x is defined by $rf_G(x) = rc_G(x)/pc_G(x)$.

Theorem 5. *For $u, v \in V$, let $p = pc_G(v) \neq 0$, $q = rc_G(v)$, $c = 1 - p - q$, $D = d_G(v)$, and $d = d_G(u,v)$. Then the hitting time h_{uv} is:*

$$h_{uv} = \begin{cases} \beta_0 \left(\lambda^D - \lambda^{D-d}\right) + \beta_1 d & \text{if } q \neq p \\ \alpha_0(d - d^2) + \alpha_1 D d & \text{if } q = p \end{cases} \quad (5)$$

where $\lambda = \frac{q}{p}$, $\beta_0 = \frac{q}{(p-q)^2}$, $\beta_1 = \frac{1}{p-q}$, $\alpha_0 = \frac{1}{2p}$, $\alpha_1 = \frac{1}{p}$.

The proof follows directly from Theorem 3 above. When $q > p$, the main determining factors in the hitting time are the regress factors $\lambda = q/p$ and D; the hitting time grows exponentially with D and polynomially, with degree D, with λ. As long as λ and D are fixed, changing other structural parameters such as the branching factor b can only increase the hitting time linearly. Note that also for $q > p$, it does not matter how close the start state is to the goal. The hitting time mainly depends on D, the largest goal distance in the graph.

Analysis of the Transport Example

Theorem 5 helps explain the experimental results in Figure 1. In this example, the plateau consists of all the states encountered before loading the package onto one of the trucks. Once the package is loaded, h_{FF} can guide the search directly towards the goal. Therefore, the exit points of the plateau are the states in which the package is loaded onto a truck. Let $m < n$ be the location of a most advanced truck in the chain. For all non-exit states of the search space, $q \leq p$ holds: there is always at least one action which progresses towards a closest exit point - move a truck from c_m to c_{m+1}. There is at most one action that regresses, in case $m > 1$ and there is only a single truck at c_m which moves to c_{m-1}, thereby reducing m.

According to Theorem 4, setting $q = p$ for all states yields an upper bound on the hitting time, since increasing the regress factor can only increase the hitting time. By Theorem 5, $-\frac{x^2}{2p} + (\frac{2D+1}{2p})x$ is an upper bound for the hitting time. If the number of trucks is multiplied by a factor M, then p will be divided by at most M, therefore the upper bound is also multiplied by at most M. The worst case runtime bound grows only linearly with the number of trucks. In contrast, systematic search methods suffer greatly from increasing the number of vehicles, since this increases the effective branching factor b. The runtime of systematic search methods such as greedy best first search, A* and IDA* typically grows as b^d when the heuristic is ineffective.

This effect can be observed in all planning problems where increasing the number of objects of a specific type does not change the regress factor. Examples are the vehicles in transportation domains such as Rovers, Logistics, Transport, and Zeno Travel, or agents which share similar functionality but do not appear in the goal, such as the satellites in the satellite domain. All of these domains contain symmetries similar to the example above, where any one of several vehicles or agents can be chosen to achieve the goal. Other examples are "decoy" objects which can not be used to reach the goal. Actions that affect only the state of such objects do not change the goal distance, so increasing the number of such objects has no effect on *rf* but can increase b. Techniques such as plan space planning, backward chaining planning, preferred operators, or explicitly detecting and dealing with symmetries can often prune such actions.

Theorem 5 suggests that if $q > p$ and the current state is close to an exit point in the plateau, then systematic search is more effective, since random walks move away from the exit with high probability. This problematic behavior of RW can be fixed to some degree by using restarting random walks.

Analysis of Restarting Random Walks

Theorem 6. *Let $G = (V, E)$ be a FSH graph, $v \in V$, $p = pc_G(v)$ and $q = rc_G(v)$. Let $R = RRW(G, s, r)$ with $0 < r < 1$. The hitting time $h_{sv} \in O\left(\beta\lambda^{d-1}\right)$, where $\lambda = \left(\frac{q}{p} + \frac{r}{p(1-r)} + 1\right)$, $\beta = \frac{q+r}{pr}$ and $d = d_G(s,v)$.*

Proof. Let $d = d_G(s,v)$. According to Theorem 1 and

Lemma 1,

$$\begin{aligned} h_0 &= 0 \\ h_x &= (1-r)\left(qh_{x+1} + ph_{x-1} + ch_x + 1\right) + rh_d \\ h_D &= (1-r)\left(ph_{D-1} + (1-p)h_D + 1\right) + rh_d \end{aligned} \tag{6}$$

Let $u_x = h_x - h_{x-1}$, then for $x < d$,

$$\begin{aligned} u_x &= (1-r)(qu_{x+1} + pu_{x-1} + cu_x) \\ &= \frac{(1-r)q}{1-c+cr}u_{x+1} + \frac{(1-r)p}{1-c+cr}u_{x-1} \end{aligned}$$

Since $\frac{(1-r)q}{1-c+cr}u_{x+1} \geq 0$ and $c = 1-p-q$,

$$\begin{aligned} u_x &\leq \frac{(1-r)p}{q(1-r)+p(1-r)+r}u_{x-1} \leq \lambda^{-1}u_{x-1} \\ u_x &\leq \lambda^{d-x}u_d \\ h_x &\leq \sum_{i=1}^{x} u_i \leq u_d \sum_{i=1}^{x} \lambda^{d-i} \leq \lambda^{d-x}\left(\frac{\lambda^x - 1}{\lambda - 1}\right)u_d \end{aligned}$$

The value u_d is the progress time from the goal distance d. Therefore,

$$u_d = (1-r)\left(cu_d + q(1+u_{d+1}+u_d) + 1\right) + ru_d$$

Since R restarts from s with probability r, $u_{d+1} \leq \frac{1}{r}$.

$$\begin{aligned} u_d &\leq \left(r + (1-r)(1-p)\right)u_d + \left(\frac{q}{r} + 1\right)(1-r) \\ &\leq \frac{q+r}{rp} \leq \beta \end{aligned}$$

Furthermore,

$$\begin{aligned} h_d &= u_d + h_{d-1} \leq \beta + \beta\lambda\left(\frac{\lambda^{d-1}-1}{\lambda-1}\right) \\ h_d &\in O\left(\beta\lambda^{d-1}\right) \end{aligned} \tag{7}$$

Therefore, by decreasing r, while λ decreases, β increases. Since the upper bound increases polynomially (the degree depends on $d(s,v)$) by λ and only linearly by β, to keep the upper bound low a small value should be chosen for r, especially when $d(s,v)$ is large. The r-value which minimizes the upper bound can be computed from Equation 7.

Comparing the values of λ in the hitting time of RW and RRW, Equations 7 and 5, the base of the exponential term for RRW exceeds the regress factor, the base of the exponential term for RW, by $\frac{r}{p(1-r)} + 1$. For small r, this is close to 1.

The main advantage of RRW over simple random walks is for small $d(s,v)$, since the exponent of the exponential term is reduced from D to $d(s,v) - 1$. Restarting is a bit wasteful when $d(s,v)$ is close to D.

A Grid Example

Figure 3 shows the results of RRW with restart rate $r \in \{0, 0.1, 0.01, 0.001\}$ in a variant of the Grid domain with an $n \times n$ grid and a robot that needs to pick up a key at location (n,n), to unlock a door at $(0,0)$. The robot can only move left, up or down, except for the top row, where it is also allowed to move right, but not up.

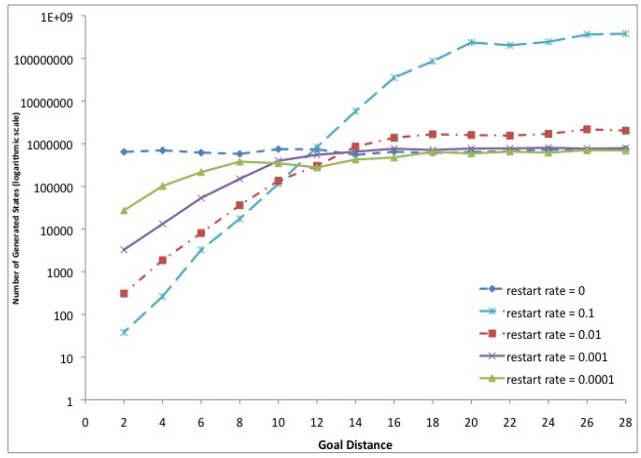

Figure 3: The Average number of generated states varying the goal distance of the starting state (x-axis) and the restart rate in the Grid domain.

In this domain, all states before the robot picks up the key share the same h_{FF} value. Figure 3 shows the average number of states generated until this subgoal is reached, with the robot starting from different goal distances plotted on the x-axis. Since the regress factors are not uniform in this domain, Theorem 6 does not apply directly. Still, comparing the results of RRW for different $r > 0$ with simple random walks where $r = 0$, the experiment confirms the high-level predictions of Theorem 6: RRW generates slightly more states than simple random walks when the initial goal distance is large, $d \geq 14$, and r is small enough. RRW is much more efficient when d is small; for example it generates three orders of magnitude fewer states for $d = 2$, $r = 0.01$.

Related Work

Random walks have been extensively studied in many different scientific fields including physics, finance and computer networking (Gkantsidis, Mihail, and Saberi 2006; Fama 1965; Qian, Nassif, and Sapatnekar 2003). Linear algebra approaches to discrete and continuous random walks are well studied (Norris 1998; Aldous and Fill 2002; Yin and Zhang 2005; Pardoux 2009). The current paper mainly uses methods for finding the hitting time of simple chains such as birth–death, and gambler chains (Norris 1998). Such solutions can be expressed easily as functions of chain features.

Properties of random walks on finite graphs have been studied extensively (Lovász 1993). One of the most relevant results is the $O(n^3)$ hitting time of a random walk in an undirected graph with n nodes (Brightwell and Winkler 1990). However, this result does not explain the strong performance of random walks in planning search spaces which grow exponentially with the number of objects. Despite the rich existing literature on random walks, the application to the analysis of random walk planning seems to be novel.

Discussion and Future Work

Important open questions about the current work are how well it models real planning problems such as IPC benchmarks, and real planning algorithms.

Relation to full planning benchmarks: Can they be described within these models in terms of bounds on their regress factor? Can the models be extended to represent the core difficulties involved in solving more planning domains? What is the structure of plateaus within their state spaces, and how do plateaus relate to the overall difficulty of solving those instances? Instances with small state spaces could be completely enumerated and such properties measured. For larger state spaces, can measurements of true goal distances be approximated by heuristic evaluation, by heuristics combined with local search, or by sampling?

Effect of search enhancements: To move from abstract, idealized algorithms towards more realistic planning algorithms, it would be interesting to study the whole spectrum starting with the basic methods studied in this paper up to state of the art planners, switching on improvements one by one and studying their effects under both RW and systematic search scenarios. For example, the RW enhancements MHA and MDA (Nakhost and Müller 2009) should be studied.

Extension to non-fair graphs: Generalize Theorem 6 to non-fair graphs, where an action can increase the goal distance by more than one. Such graphs can be used to model planning problems with dead ends.

Hybrid methods: Develop theoretical models for methods that combine random walks with using memory and systematic search such as (Lu et al. 2011; Xie, Nakhost, and Müller 2012).

References

Aldous, D., and Fill, J. 2002. *Reversible Markov Chains and Random Walks on Graphs*. University of California, Berkeley, Department of Statistics.

Brightwell, G., and Winkler, P. 1990. Maximum hitting time for random walks on graphs. *Random Struct. Algorithms* 1:263–276.

Cazenave, T. 2009. Nested Monte-Carlo search. In *IJCAI*, 456–461.

Coles, A.; Fox, M.; and Smith, A. 2007. A new local-search algorithm for forward-chaining planning. In *Proc. ICAPS'07*, 89–96.

Fama, E. F. 1965. Random walks in stock-market prices. *Financial Analysts Journal* 21:55–59.

Finnsson, H., and Björnsson, Y. 2008. Simulation-based approach to General Game Playing. In *AAAI*, 259–264.

García-Olaya, A.; Jiménez, S.; and Linares López, C., eds. 2011. *The 2011 International Planning Competition*. Universidad Carlos III de Madrid.

Gelly, S., and Silver, D. 2008. Achieving master level play in 9 x 9 computer Go. In *AAAI*, 1537–1540.

Gkantsidis, C.; Mihail, M.; and Saberi, A. 2006. Random walks in peer-to-peer networks: algorithms and evaluation. *Perform. Eval.* 63:241–263.

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

Hoos, H., and Stützle, T. 2004. *Stochastic Local Search: Foundations & Applications*. Morgan Kaufmann.

Lovász, L. 1993. Random walks on graphs: A survey. *Combinatorics, Paul Erdős is Eighty* 2(1):1–46.

Lu, Q.; Xu, Y.; Huang, R.; and Chen, Y. 2011. The Roamer planner random-walk assisted best-first search. In García-Olaya et al. (2011), 73–76.

Nakhost, H., and Müller, M. 2009. Monte-Carlo exploration for deterministic planning. In *IJCAI*, 1766–1771.

Nakhost, H.; Hoffmann, J.; and Müller, M. 2012. Resource-constrained planning: A Monte Carlo random walk approach. Accepted for ICAPS.

Norris, J. R. 1998. *Markov chains*. Cambridge University Press.

Pardoux, É. 2009. *Markov processes and applications: algorithms, networks, genome and finance*. Wiley/Dunod.

Pham, D. N.; Thornton, J.; Gretton, C.; and Sattar, A. 2008. Combining adaptive and dynamic local search for satisfiability. *JSAT* 4(2-4):149–172.

Qian, H.; Nassif, S. R.; and Sapatnekar, S. S. 2003. Random walks in a supply network. In *40th annual Design Automation Conference*, 93–98.

Selman, B.; Levesque, H.; and Mitchell, D. 1992. A new method for solving hard satisfiability problems. In *AAAI*, 440–446.

Valenzano, R.; Nakhost, H.; Müller, M.; Schaeffer, J.; and Sturtevant, N. 2011. ArvandHerd: Parallel planning with a portfolio. In García-Olaya et al. (2011), 113–116.

Xie, F.; Nakhost, H.; and Müller, M. 2012. Planning via random walk-driven local search. Accepted for ICAPS.

Yin, G., and Zhang, Q. 2005. *Discrete-time Markov chains: two-time-scale methods and applications*. Springer.