倉庫番ゴールエリアの自動配置手法

小田原大　金子知適　川合慧
{haradai, kaneko, kawai}@graco.c.u-tokyo.ac.jp
東京大学大学院 総合文化研究科

本研究では、予め手順を用意して通路を生成し、通路の不要なスペースを削除するという手法を用いて、倉庫番におけるゴールエリアの自動配置を行なった。手順が最初から用意されているため、配置された問題に関しては解をもつかどうかを確認する必要がない。結果として、回り道を繰り返さなければ攻略できない配置を作成することができた。

A Method of Automatic Creation of Goal-Area in Sokoban Maps

Masaru Odawara　Tomoyuki Kaneko　Satoru Kawai
Graduate School of Arts and Sciences, The University of Tokyo

A new method fo automatic creation of goal-areas in sokoban game is proposed. The method consists of two phases. In the first phase, the path of a packet and that of the keeper are determined incrementally one packet by one. In the second phase, the set of free spaces, obtained in the first phase, is shrunk as much as possible, preserving the feasibility of solutions. Some experiments showed that reasonably difficult sokoban goal-areas can be generated by our method.

1　はじめに

1.1　倉庫番について

倉庫番とは、マップ上に置かれた荷物をゴールへ全て運び込むというパズルゲームである。番人は荷物を押して運ぶことができる。二つ以上の荷物を同時に押すことはできず、引くこともできない。このような単純な制限しかないが、荷物を全てきちんとゴールに納めるためには複雑な手順が要求される奥の深いゲームである。

問題のマップは図1のようになっている。番人が一人いて、スペース（通路）上に荷物が配置されている。この荷物を納めるゴールが図の左側にある。倉庫番ではゴールはこのように密集して配置されている問題が多い。このようにゴールが密集している辺りの地形を指して、ゴールエリアということにする。

1.2　ゴール収納の難しさ

倉庫番には、(i) マップを埋める荷物のどれかを動かして通路を空け、どのような順で運び始めるか、(ii) 荷物をどの順にゴールエリアに収納していくか、

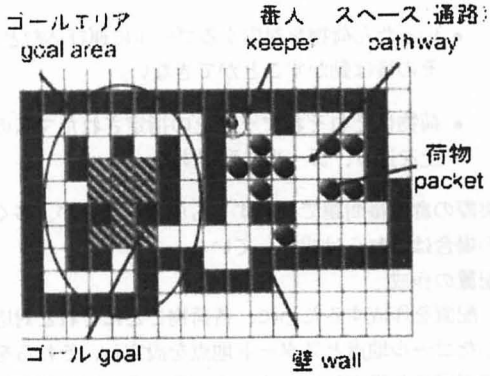

図 1: マップと地形 (XSokoban[1] level 86 より)

という2つの難しさがある。

例えば図2は XSokoban の level.44 の全体マップだが、マップ左側の部分と右側のゴールエリア部分とがほぼ独立しており、(i)、(ii) の難しさを備えた問題である。ゴールへ荷物を全て格納するためには手順を凝らす必要があり、簡単ではない。

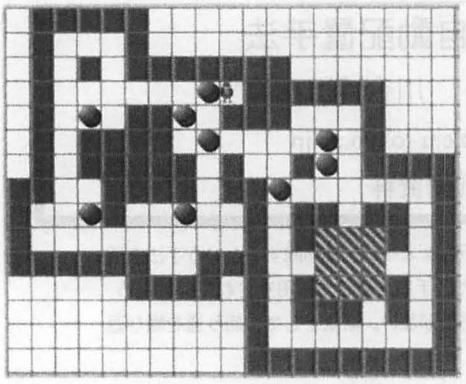

図 2: xsokoban Lv.44

1.3 自動配置のアプローチ

本研究は、(ii) の難しさに着目し、ゴールエリアを自動生成することを目標とする。

本研究の対象

ゴールエリアの自動生成に際して以下の前提をおいた。

- 個々の荷物とゴールの対応は決まっている。
- いったん荷物を対応するゴールに運び込むと、その後は動かすことができない。
- 荷物はそれぞれマップ上の指定されたマスの上を通り、ゴールに到達する。

実際の倉庫番問題では必ずしも成立しないが、多くの場合はこれらは成立している。

配置の作成

配置を作成するために、各荷物ごとにそれと対応したゴール地点とスタート地点を設定し、それらを結ぶパスを設定していく。その際、番人が荷物を押すために必要なスペースも用意する。一度ゴールに配置した荷物は、番人も荷物もその上を通ることのできない障害物とみなされる。

この結果生成した配置に対しては、必ず全ての荷物を順にゴールに格納することができる。

1.4 先行研究との比較、関連研究

倉庫番の問題の自動作成については村瀬 [2] がある。ここでは、全体の問題の大きさが 8x8、荷物の数が 3 つのものに限定して生成しているが、これに対して本手法ではゴールエリアのみで 12x12 程度[1]、荷物の数は 10 以上のものまで簡単に生成することができる。また、本研究では解が存在するように構成的にマップを生成するため、生成された問題は必ず解を持ち、解の存在について solver によって確認する必要がない。また荷物を格納する順序も配置の生成の際に決めており、容易に確認することができる。

関連研究として、上野、中山、疋田 [4]、Andreas[5]、高橋 [7] などが倉庫番の solver を開発している。

2 自動配置アルゴリズム

ゴールエリアを配置するアルゴリズムの概略を以下に示す。

2.1 用語の定義

最初に、本稿で用いる用語について説明する。

マス 構成単位。この上に地形が配置される。マスを指してマップ上の点という。

地形 (または 要素) マスの状態。壁、荷物、番人、ゴール、障害物、スペース (通路) のいずれかを指す。

マップ 地形が配置されたマスが格子状に並んでいるもの。

障害物 自動配置手法の実行過程において、番人も荷物もその上を通ることができない地形を表す。

パス 荷物の移動の順路。

コーナー パスにおける曲り角を意味する。普通は直角の曲り角である。パス n 上の k 番目 ($k \geq 0$) のコーナーを C_{nk} と表す。

バイパス コーナーにおいて、番人が荷物を押す方向を変えるために通過する通路。コーナー C に対するバイパスの始点を $in(C)$、終点を $out(C)$ で表し、バイパス自体を $in(C)$〜$out(C)$ と表す。

一時スペース 最初にバイパス用のスペースとして設けられるスペース。(図 5)

[1] 計算時間をかければマップについてはどんな大きさでも生成できる

スタート地点/ゴール地点 荷物が最初に通るべき点と、荷物を納めるべきマス。ゴール地点がその荷物のゴールとなる。n 番目の荷物について、S_n, G_n と表す。スタート地点については重複は許されるが、ゴール地点は異なる n について重複してはならない。

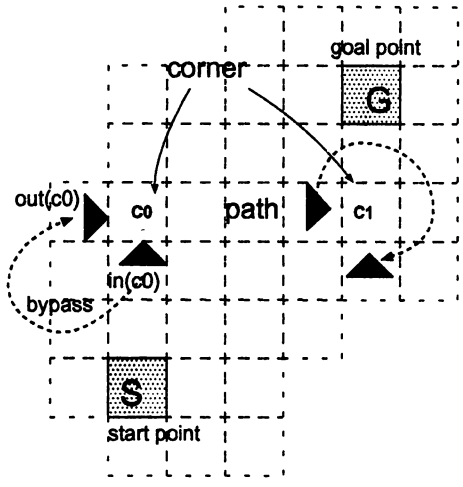

図 3: 本稿で用いる用語

2.2 ゴールエリアの構成

まず壁だけからなるマップを考える。このマップ上に荷物 n ($n = 0, 1, 2, ..., m-1$; m = 荷物の個数) に対応するゴール G_n と、スタート地点 S_n を設定し、$n = 0, 1, 2...$ の順に、S_n と G_n を結ぶパスを可能なパスから選ぶ。それから、パスに必要となるスペースとバイパス用のスペースを用意する。これにより荷物 n をスタート地点からそのゴール地点まで必ず運ぶことのできるマップが生成される。これを m 回、$n = m-1$ になるまで繰り返し、最終的なマップを生成する。

$n > 0$ のとき、荷物 n のスタート地点からゴール地点のパス上に 0 以上 n 未満のゴールが設定されている可能性がある。そのようなゴールには荷物が既に納められていることになるため、パスはそのような荷物の上を通ることができない。

なお荷物 n を設置したあとで、番人が S_{n+1} (荷物 $n+1$ のスタート地点) へと移動できなければならない。ここでは前提として、全 m 個の荷物に関して設定する間は常に、各スタート地点を互いに行き来できるとした。

全ての荷物に関してスタート地点、ゴール地点とパスを設定した後、不要なスペースの削減を行なう。このことをスペースの縮約ということにする。詳細は 3 章で後述する。

基本構成手法

1. m 個の荷物について全て、スタート地点とゴール地点を適当に決め、$n = 0$ とする。
2. 全ての荷物 $t(t = 0, 1, \cdots, m-1)$ について以下を繰り返す。
   (a) 既存の荷物や障害物を考慮して、荷物 n に対応するパス n を設定する。
   (b) パス t 上の各コーナー C_{tk} について、$in(C_{tk})$ と $out(C_{tk})$ を定め、番人の通路 (バイパス) を設定する。
   (c) 設定されたパスとバイパスに関してスペースを設定する。(図 5)
3. 各パスに関して不要なスペースを削減する。

図 4: 基本構成手法

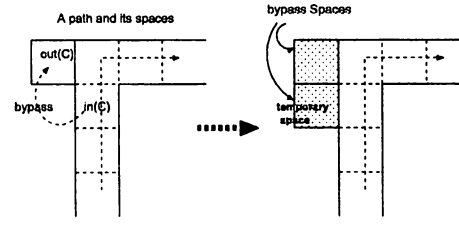

図 5: 番人の移動に必要なバイパスとバイパススペースの設置

2.3 パスの設定

パスを選択するために、まず可能なパスの集合を求める。

可能なパスとその選択

番人がパスにしたがって荷物をゴール地点まで運ぶためには、以下の条件を満たす必要がある。

- パス上に荷物または障害物が置かれていてはならない。…(1)

- 最初に設定するバイパス上の一時スペースに当たる部分に荷物または障害物が置かれていてはならない。…(2)

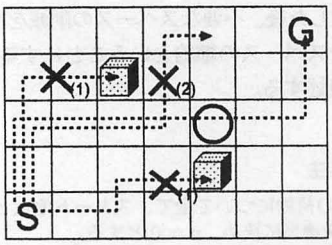

図 6: 可能なパス（○）と不可能なパス（×）

パスの選択

上の条件を満たすパスから、一通りをランダムに選び、その荷物のパスとする。

3 スペース効率の最適化

3.1 スペースの縮約

上の構成手法に基づき、m 個の荷物についてパスを設定した段階では（図 4-3 の直前）、例えば図 7 のように、必要以上に多くのスペースが設定されてしまい、何通りもの解法が可能となる「つまらない」配置となってしまう。そこで、パスを全て設置し終わった後で、図 4-3 において不要なスペースを削除する。

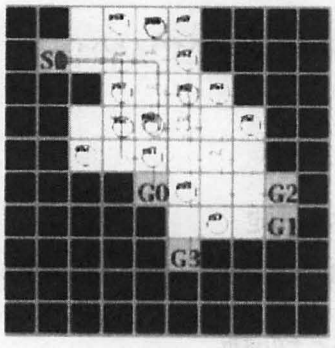

図 7: スペースが広過ぎるゴールエリア例

荷物のパスに対応するスペースは削除することはできないが、最初に設定するとしたバイパス用の一時スペースについては、$in(C)$ と $out(C)$ の間を結ぶ通路が図 5 の一時スペース以外に存在することを確かめられれば、これは不要となり、壁に戻すことができる。これをスペースの縮約と呼ぶ。縮約を可能な限り行なうことで、余計なスペースが減るため、スペースがより効率的に用いられることになり、結果として難しい問題を生成できると考えられる。

縮約を行なうためには、図 8 左の状態から右の状態へと移行可能であればよい。そのためには適切な既存のスペースが必要であり、本研究では以下の 2 通りの手法によってこれを確かめる。

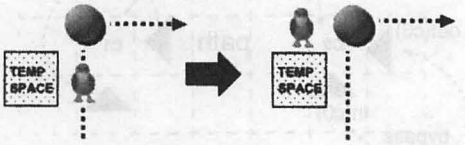

図 8: 一時スペース削減の条件 図左の状態から図右の状態へ、一時スペース上を通らずに移行することができれば、縮約が可能である。

縮約その 1 基本構成手法の 2b で設定したバイパス用スペース以外に $in(C_{nk})$ から $out(C_{nk})$ へ至るスペースを探す。存在していれば、一時スペースを縮約候補とする。図 9 は縮約その 1 が可能な一例である。図にあるグレーの四角が縮約候補となる一時スペースである。

縮約その 2 図 10 のように、破線のパスから太い実線のパスへとパス自体を拡張し、新しいコーナーに新たな in, out を対応させる[2]。図 10 では、破線のパス上のコーナー C_0 に対応する $in(C_0)$ と $out(C_0)$ に関して、実線のパス上のコーナー C_1, C_2 に対する $in(C_1)$, $out(C_1)$ と $in(C_2)$, $out(C_2)$ で置き換えている。ここで、バイパス $in(C_1)$〜$out(C_1)$、バイパス $in(C_2)$〜$out(C_2)$ が存在しうる場合、一時スペースを縮約の候補とする。

図 11 は縮約その 2 が適用される例である。他にも既存のスペースの形がたとえば図 12 のようなものであれば、破線のパスから実線のパスへとパスを拡張することにより、縮約その 2

[2]パスの拡張を考えれば、より多様な縮約を考えることができる。今回の研究においては、パスの拡張はこれ以上は行なっていない

が可能である。これらの図においても、グレーの四角が縮約候補の一時スペースを表す。

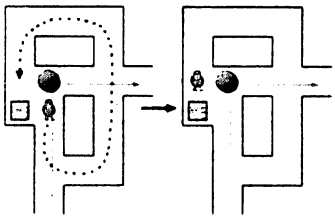

図 9: 縮約その 1 が可能になる例

3.2 縮約の適用

縮約候補を求めたが、これらのスペースを全て削除してしまうと他の荷物に必要なパス上のスペースやバイパス用のスペースまで削除されることがある。これを防ぐため、全ての縮約を行なった後で必要なスペースまで削除されていないかどうかのチェックを行なうことで解の存在を保証する。

3.3 縮約確認の順序

縮約可能性の確認はその順序に依存する。例えば、パス 1 →パス 2 →パス 3 と縮約した場合と、パス 3 →パス 2 →パス 1 の順に行なった場合では、結果として異なる縮約が行なわれ、削除されるスペースの数が異なることがある。よりスペースの数を少なくするために、最適な順序を探索する必要がある。

3.4 ゴールによるスペースの充填

上述のアルゴリズムを適用し得られた、ゴール以外のスペースに関してゴールとして利用可能なものを選ぶことができる。マップの広さに比して多くのゴールを設置することができ、みかけの難しさは向上する。

スペースの充填はマップが決定した後（図 4 の後）に以下の手順で行なう。pは荷物に対応する番号とし、m は荷物の総数とする。パス $p(p = 0, 1, \cdots, m-1)$ について以下を繰り返す。

1. パス p について、p 以上のパスもしくはそれに伴うバイパスが通過するかどうかを確認する。

2. パス p 上の他のパス/バイパスが通過しない点までを G_p に近い方から順にゴール地点とする。

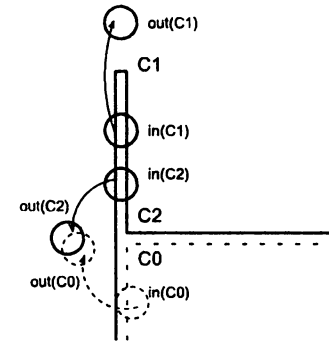

図 10: 縮約その 2

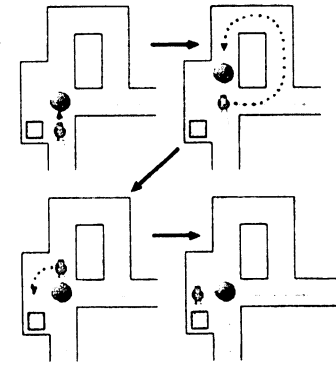

図 11: 縮約その 2 が可能になる例

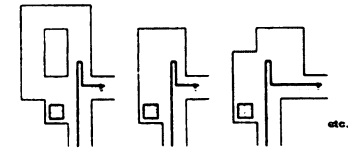

図 12: 縮約 2 が可能となるスペースの例

4 問題の生成

基本構成手法とゴールの充填手法を用いてゴールエリアを自動配置実験を行なった。

4.1 生成の条件

スタート地点の配置 マップの幅を $w \times h$ とする。(w, h はそれぞれ、マップの x 軸方向、y 軸方向の幅を表す整数値) このとき x 座標が 2 または $w-1$, y 座標が 2 または $h-1$ となるようにスタート地点の位置に制限を加え、その範囲内で重複を許してランダムに設定する。

ゴール地点の配置 マップ内でスタート地点よりも設定するマップの内側に含まれるようにし、重複を防ぎつつランダムに設定する。

障害物の設定 スタート地点の x 座標が 2 か $w-1$ の場合にはその上下のマスに、y 座標が 2 か $h-1$ の場合にはその左右のマスに障害物を設定した。

マップの広さ 10x10 から 15x15 の間で設定した。

荷物の個数 荷物は 4 個から 8 個程度の間で設定した。

縮約の順序探索 荷物の数が少ないため、全列挙して最適な縮約を求める。

4.2 結果

条件にしたがって生成された結果の例を図 13 から図 16 に挙げる。図 13、図 15 が生成された図、これにゴールの充填を適用し、適当に荷物を配置して一つの問題としたものが図 14、図 16 である。

4.3 評価

文献 [2] においては、マップの広さに対する手数、持ち替えの回数、遠回りの三つの要素を面白さの基準としていた。ここで作成した配置については、縮約を一度行なうたびに必ず一度は荷物を押すために回り道をしなければならない。縮約の回数が配置の複雑さの一つの指標となっていると考えられる。

マップの広さ 10x10 の広さの中に 4 つのゴールを持つマップを生成し縮約がどのように行なわれているかを確かめる。

スタート地点、ゴール地点については前述の生成条件にしたがう。約 200 回、同一条件で自動的に生成し、スペースの数と縮約の回数について調べ、平均値をとった。(表 1)

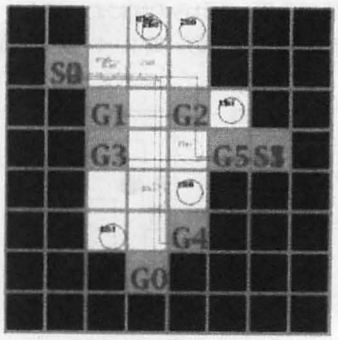

図 13: 生成した図 2

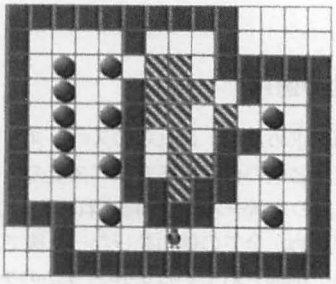

図 14: 図を用いた問題例

	平均
縮約回数	1.41
スペースの数	26.71

表 1: 縮約回数と最終的に生成されたスペースの個数

縮約回数の分布は図 17 にある。最大 4 つの一時スペースが縮約された。

アルゴリズムの可能性

xsokoban の問題について、スタート地点とゴール地点を適切に定めれば問題が作成できるかを確認する。

特徴的な例として再び図 2 のゴールエリアの構成を例にとる。9 個のゴールエリアは図 18 の右図のように正方形に配置されている。

各ゴールエリアに対し、スタート地点とパスを図 18 の左図のように配置することによって、問題と同じゴールエリアを構成することができる。

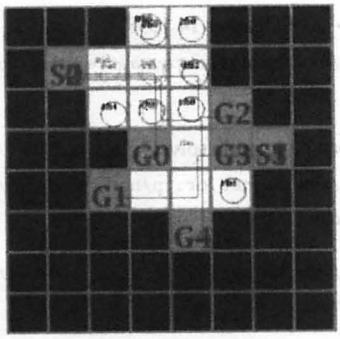

図 15: 生成した図 3

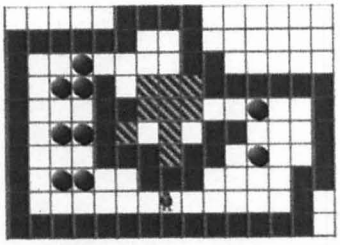

図 16: 図を用いた問題例

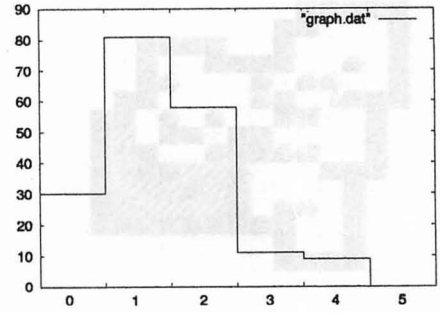

図 17: 縮約の回数（縦軸：生成された配置の個数、横軸：縮約の回数）

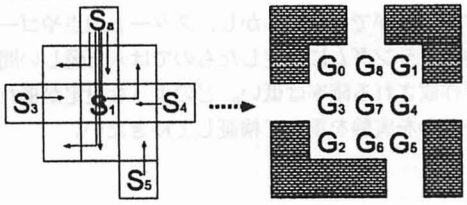

図 18: level 44 の構成 (S_α は S_0,S_2, S_6,S_7,S_8 を表す)

5 今後の課題

5.1 生成困難な問題

xsokoban の Level 16(図 19) は高橋の *"Sokoban Automatic Solver"*[8]、Andreas *"Rolling Stone"*[6] によっても解けない「難問」[3]だが、この問題の難しさには、一つには最初にあげた倉庫番の難しさの要素が (i) と (ii) に明確に分かれておらず、ゴールへ格納する順序と、ゴールエリア以外の荷物の移動の順序が独立していないことにあると思われる。

現在このような問題を生成するのは本稿で示した手法では難しい。

5.2 縮約の改良

パスの拡張

最初に設定されたパスを変更しながら縮約を行なう方法には、本稿で提案した以外にもいろいろな手法が考えられる。

スペース効率の向上

縮約に必要なスペースと不要なスペースが混在し、このアルゴリズムを直接実現しようとすると、バイパス用のスペースまで縮約の対象としてしまうため、これを防ぐチェックが不可欠となっている。最初に設定するスペースはパス用のスペースのみとし、番人の移動に最終的に必要になるものだけをあとから設定することにすれば、余計な縮約は行なわれない上、よりスペース効率のよいゴールエリアの配置が可能であると考えられる。

中間地点の設置

スタート地点から直接ゴール地点に至るのではなく、途中で通過する別の地点を定めるとより複雑な問題が生成できると思われる。

6 まとめ

縮約を用いた配置自動生成アルゴリズムは、人為的にスタート地点の位置等の要素を設定し問題を構成すればかなり難解な問題でも作成可能であること

[3]xsokoban 全 90 問については高橋の *Automatic solver* は 78 問、andreas の *Rolling Stone* は 52 問 (ウェブサイト [6] の記載による) を解くことができる。*Automatic Solver* で解けず *Rolling Stone* で解ける問題は level.40 と level.75 の 2 問である

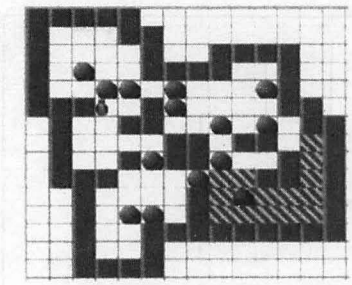

図 19: xsokoban Lv.16

がわかった。また、回り道を繰返し通らなければ荷物をゴールに納めることができないような配置を作成することができた。しかし、スタート地点やゴール地点をランダムに設定したものでは未だ難しい問題が作成される確率は低い。どのような設定が優れているかを実験を重ねて検証してゆきたい。

謝辞

田中研ゼミ (GPS) の参加者の方々のアドバイスに感謝致します。

References

[1] XSokoban Home Page
http://www.cs.cornell.edu/andru/xsokoban.html

[2] 村瀬芳生、松原仁、平賀譲「倉庫番」の問題の自動作成. 情報処理学会論文誌 Vol.39 No.03-007, pp.567-574, 1998.

[3] Y. Murase, H. Matsubara, and Y. Hiraga. Automatic making of sokoban problems. *In Pacific Rim Conference on AI*, pp.592-600, 1996.
http://citeseer.nj.nec.com/murase96automatic.html

[4] 上野篤、中山康、疋田輝雄 第3章 倉庫番 松原仁・竹内郁雄編 ゲームプログラミング pp.158-172, 共立出版, 1998.

[5] Sokoban Homepage
http://www.cs.ualberta.ca/%7egames/Sokoban/

[6] Status - as of October 14 1998
http://www.cs.ualberta.ca/%7egames/Sokoban/status.html

[7] "Sokoban Automatic solver" for windows
http://www.ic-net.or.jp/home/takaken/e/soko/index.html

[8] http://www.ic-net.or.jp/home/takaken/e/soko/x70.txt