自动扫雷程序：实现原理，误区，辨析与展望

摘　要：自动扫雷程序，或称扫雷 AI，指能够自动识别扫雷游戏或自身生成一个局面，经逻辑演算，而非读取内存等作弊方式，得出每一格是否可打开或标雷，或关于概率的结论，进而不断自动点击推进局面，并输出数据的程序。近年来，自动扫雷程序再次成为研究热点，多位学者编写了不同的自动扫雷程序。然而其中大多存在不规范，大多数工作属于“重复造轮子”，缺乏创新性。为避免后人重蹈覆辙，规范研究，本文将从实现原理，误区，辨析与展望，对自动扫雷程序研究进展综述如下。

关键词：自动扫雷程序；扫雷 AI；自动解局；算法；扫雷局面学。

自动扫雷程序最早可以追溯到 2005 年，由 Mathias 写的 Minesweeper Solver[1]。软件可以自动识别系统自带的扫雷游戏（XP 及之前），经后台逻辑演算后给予玩家游戏提示，或不断自动点击推进局面，高级游戏胜率可达到 36.40±0.02%。遗憾的是，作者并未开源或详述其原理。近年来，自动扫雷程序再次成为研究热点。多位学者编写了不同的自动扫雷程序，然而其中大多或多或少存在一些问题，存在不严谨之处。本文将从实现原理，误区，辨析与展望，对自动扫雷程序相关研究进展综述如下。

1. 实现原理

本文所讨论的自动扫雷程序指模拟人的行为，对局面进行逻辑推演分析后决策或采取行动的程序，通过读取内存等作弊方法不是本文的讨论对象。所有的自动扫雷程序均可大致分为以下四个步骤：获取局面、逻辑推理、猜雷、采取行动。

1.1 获取局面

获取局面是一切自动扫雷程序的开始，它有两种实现方式：一是通过图像处理识别的方法，从打开的扫雷软件中直接读取当前局面，例如 Minesweeper Solver[1]即可自动识别 win 自带扫雷并进行后续分析，Vespa 的文献[2]中对这一图像识别的过程进行了详述；二是直接由自动扫雷程序随机生成一个局面，然后

再自己来解。前者目前看来具有较强特异性，xp 自带扫雷，Arbiter，Clone 等软件界面几乎完全相同，然而程序只能在其中某一款中才能运行。后者则需注意随机算法的合理性，详见本文 2.1。

1.2 逻辑推理

获取局面后下一步即为逻辑推理。这一步的目标是找出所有一定能打开的格子。然而限于不同学者的计算策略，不少人的算法可以找出一定可以打开的格子，但做不到把所有能打开的格子找全。需要明确的是，扫雷求解是 NPC 问题而不是 P 问题，这已是数学界共识[3]，只有使用枚举罗列出所有情况才能把解找全，使用 P 方法注定会漏解。然而由于枚举的费时，因此建议将逻辑推理这一步分层，先从简单推理/定式/减法等下手，等前者无能为力时，再采用枚举求解。

1.21 简单推理

即指当数字等于周围未打开格数时标记，当数字周围标记数等于其本身时，将其他格子打开，最基本的策略。

1.22 定式/减法公式

减法公式亦称双集合模型[4]。绝大部分扫雷的定式，例如边 11,12 公式，等等，都是基于双集合模型的，特点是只要对两个相邻的数字及其周围格子推演，就能推出哪格一定可以打开或标记，这一过程不涉及到 3 个或更多数字的相互纠缠。然而在普通扫雷中，减法固然可以解决 90%以上的问题，但并不是 100%，而且也有减法由并非相邻的两个数字构成。因此仅使用减法及基于减法的定式并不能找出所有可以打开的格子，然而遗憾的是不少学者即止步于此。

1.23 高斯消元求解

高斯消元是一种讨巧的 P 方法。原理是将扫雷每一个未知的格子看成未知数，每一个已知的数字即为一个方程联立线性方程组[5]，对线性方程组的增广矩阵高斯消元求解。但方程组是欠定的，理论上有无数组解，我们也难以找出每一个未知数仅为 0 或 1 的解。王嘉宁及王泽尧在他们的算法中通过遍历矩阵的行最简式的每一行，通过必要条件找出一定为雷或不为雷的格子，具体过程可见[6,7]。需要说明的是，前者对这种方法寻找固定解的充分性及必要性进行了证明，但必要性的证明有误，因此这种讨巧的 P 方法只能作为 1.22 策略的补充，可以找出

1.22 策略中未找出的解，但仍做不到 100%找出所有能打开的格子。

1.24 枚举+优化

只有枚举才能做到完美的逻辑推理，同时枚举的结果有助于 1.3 猜雷策略中的概率计算。但枚举十分消耗算力，假设高级 400 格则有 2 的 400 次方种可能，即使计算机也无法胜任，因此需要优化。

优化首先需区分前沿区和空白区[8]，只需对前沿区进行枚举即可。在这基础上，王泽尧采用了 DPS+剪枝的方法进行优化，在保证准确的基础上获得了良好的运算速度[9]。（参考文献中没有提到过多原理，仅作为研究链接给出）

在枚举的过程中，必须注意剩余雷数带来的影响，将剩余雷数不符的情形剪除。

1.3 猜雷

在确定没有一定可以打开的格子后下一步就是猜雷。猜雷的策略有基于概率最小、随机点一个、猜角、找数字 1 周围格子最多的下手等等。这边对基于概率最小进行详述。

如何结合剩余雷数信息进行最准确的概率计算看似简单，实际较为复杂。忽略剩余雷数信息的算法绝不能说是完美的算法。具体数学过程可见[5,10]，大致分为前沿区枚举、空白区加权、计算每格概率三步。概率计算的具体方法可以不同，但原理都是一致的，除此之外的概率计算原理都是错误的。由此，算法可以得出每一格为雷的具体概率，并选择概率最小的方块点击。

当有多个方块概率相同时如何选择，详见 2.7。

1.4 采取决策行动

根据 1.2 的结果或 1.3 的猜雷策略进行相应的操作，例如模拟鼠标对扫雷程序点击、标雷、双击；或输出结果供玩家参考；若局面由自动扫雷程序生成，则对模拟盘面进行相应操作。重复上述步骤直至游戏结束，或设置自动重开，多局后输出胜率等结果。

本节只是阐述了大致思路和步骤，从 1.1 到 1.4，每一步都存在大量的具体

优化措施及细节问题，因笔者水平所限，不作展开。

2. 细节及误区

2.1 重视局面生成的随机性

扫雷的随机性十分容易被忽视，合理的随机算法涉及随机数底层机制，远不止“480 选 99”这么简单。XP 及之前的扫雷，就因为忽略了随机算法的随机而导致了著名的“局面循环”重大 bug[11,12]；也有研究提示 win7 扫雷和 Arbiter 在死猜的特征上存在差异[13]，可以认为是随机算法的不同导致。Arbiter 会对 3BV 过小的局面过滤，而自制的随机算法大部分不会。因此随机算法的不同，即会导致研究结果（如极限胜率，平均 3BV 等）的差异。

为了以防被加以利用，目前 Arbiter 的随机算法是保密的。虽然没有证据显示 Arbiter 随机算法最为合理，但它经过了多年的实战检验，相比其他算法可以认为随机性最好。因此推荐在 1.1 的过程中优先采用 Arbiter 图像处理。诚然，采用图像处理较为费时，自制局面仅需数毫秒即可完成而图像处理+模拟点击则需数秒，因此一些大样本的情况下也必须采用后者。若采用自制的随机算法，则建议说明随机机制。

2.2 胜率等数据需要带上样本量及标准差

很多研究者通过自动扫雷程序研究极限胜率，但在给出的数据中不注明样本量或标准差。这边给出标准差公式：$S_P=\sqrt{\frac{P(1-P)}{n}}$。例如 100 局赢了 38 局，则 $S_P=\sqrt{\frac{0.38(1-0.38)}{100}}=0.049$，胜率记为 38±4.9%。标准差代表抽样误差的大小。在高级游戏的极限胜率计算中，只有约 2500 局以上才能保证±1%，250000 局以上才能保证±0.1%，因此建议跑的局数至少要达到 2500，否则意义不大，也没法互相比较。

2.3 古典概型是否适用？

文献[9]中认为，对于死猜，虽然古典概型认为概率相同，但我们可以采用“雷在排布时更倾向于平均分布”，通过对局面四分统计雷数从而认为哪一种可能更大。类似观点还有“数死猜每一行或列哪行雷更少，那行就是雷”。这些观点是

站不住脚的。只要保证布雷是完全随机的，那么扫雷概率计算完全适用于古典概型，这是条件概率的计算，点开的数字就是先决条件。就好像“扔硬币前十次都是正面并不会使第十一次反面几率更大”，具体过程不作展开。

然而目前并没有任何一种随机算法可以保证是真随机，将来也不会有，因此相关的死猜概率验证的工作并不是徒劳，仍有一定的意义。

2.4 基于概率最小是否就等同于胜率最大？

是不是点击当前概率最小的格子就一定等同于最终胜率最大？没有这个保证，也没有任何相关研究。的确可能冒一下险点击当下概率不是最小的格子，但成功打开后对后续帮助会比概率最小的格子更大。目前而言没有什么好的研究方法，只能先把基于最小概率的自动扫雷算法弄成熟，再把猜雷策略换成其他策略加以验证分析。

2.5 缺乏可验证性

可验证性是衡量一个研究是否可靠最为重要的依据。遗憾的是大部分研究都欠缺。可验证性最直接的证据是公开代码[14]或人性化的工具[1]，其次是关于原理细节的说明及效果视频，最差的是只公布几个数据，其他完全不提[15]。笔者在和几位作者交流的过程中，每次程序第一次发表时都会存在不少 bug，笔者观看视频或动图后再告知作者改进，不断完善。因此如果没有视频等验证方式，作者可能会“将错就错”；或为了标榜自己算法先进，虚报胜率；以及部分过于复杂而卡死或直接重开的局如何处理（直接剔除是不合适的），这些都会影响到最终的结果。建议作者公开代码或者提供人性化界面的工具，供读者自行检验。

2.6 第一点的选择

文献[15]指出，在第一点不开空情况下，角开胜率最高，达 38.2%；在第一点必开空情况下，中间开胜率最高，达 49.6%。该研究鉴于可验证性及许多未提及的细节，具体胜率数字不一定准确，但大致数字及这个结论具有参考意义。因此追求胜率的自动扫雷程序第一点一定选择角（若第一点必开空则选中间）。

2.7 猜雷逻辑

在多个概率同样最小的格子，程序会如何选择，需要作者进行说明。这种情况多见于：①角开没开空；②三猜一死猜（特殊的死猜，虽然概率相等，但却需要选择两边的格子而不是当中的[16]）；③前沿区概率均高于空白区。不同的选择策略会影响最终胜率，作者需要进行说明。

3. 展望

3.1 每一步的模块化，建立成熟的公认算法

目前大部分作者做的工作，都属于“重复造轮子”。例如图像识别这一步，就有多位作者分别写了。逻辑推理部分大量作者分别写了各自的算法，互相之间有相同有不同，算法的准确性及优化程度上也不一致。因此建议对于图像识别/布雷的随机算法，简单推理/定式减法/枚举/概率计算等每一步过程，建立公认的模块化的算法，以便后人简便使用。使用相同的布雷算法后，可以使相互之间的结果更有可比性；逻辑推理及猜雷模块化后，可以避免做重复劳动。

3.2 极限胜率及局面指标的探索

极限胜率指在最优化策略下胜率的极限值，是大多数作者的主要探索目标。由于获取局面及猜雷逻辑和其他细节上的不同，极限胜率从 36-39%不等[1,9,14,15]。根据笔者私下交流，王泽尧的研究[9]可信度最高，极限胜率达 39.12±0.15%。极限胜率必须保证寻找固定解时基于枚举的，找到了所有的解，而不是仅基于减法或高斯消元等 P 方法；猜雷是严格按照最小概率猜的，并结合了剩余雷数信息，否则就不能称为极限胜率。

除极限胜率外，还可研究无猜胜率，采用不同逻辑层次下的胜率，遇死猜概率，不同第一点等预先条件下的胜率等等，文献[14]给出了许多很有意思的结果，可作参考。

3.3 扫雷助手工具

哪里可以开，哪里是雷，当没有格子看得出来时候我怎么猜才能概率最大化，这是困扰新手玩家最大的问题。基于图像处理的自动扫雷程序外加一个人性化操作界面可以完美解决这些问题，识别局面后给出提示（哪里一定可以开，一定可以标，以及每格概率），帮助真人的游戏过程，同时对刷自定义会有莫大的帮助。

王泽尧[9]表示不打算公开代码，但会公开这样类似的助手工具，让我们拭目以待。

3.4 AI 竞速及伦理

扫雷 AI 可以互相比胜率，比 IOE，比优化程度及时间复杂度，但互相比最快记录没有任何意义。有人认为发展扫雷 AI 会扰乱真人排行榜秩序，其实大可不必。AI 的模拟点击不存在轨迹，每一点都在方块中间，鼠标按下弹起不需时间，判雷思路和真人迥异，还有很多间接证据，一眼就能看出，因此关于扰乱排行榜的伦理担忧大可不必，作者们可放心的公开代码或工具。

有人认为 AI 如此强大，真人无论胜率还是开率还是最快记录都不及电脑，感叹“扫雷已死”。扫雷只是个游戏，只要在这个游戏过程中感受到快乐，那它就不会死，电脑只是辅助我们的工具而已。

有没有可能基于真人的思路，建一个模拟真人的扫雷 AI 呢？限制点速及移速，图像识别过程中人为加入 0.2-0.3s 作为反应时间，限制逻辑推理在固定定式范围内等等。不敢说不可能，但至少目前来看完全没有眉目。如果真做出来的话，对于验证扫雷方法，一些局面下的策略选择等等，都十分有意义，相关研究应该得到鼓励和提倡，而不应惧于伦理风险而抛弃。

3.5 破空概率计算与刷 IOE

目前已有多位作者做出了计算每一格概率的算法，但迄今为止尚未有每一格破空概率（每一格格子点下去可以破空的概率，不是 Thrp）的算法。空即为该格本身及其周围一圈都不是雷的情形，但概率并不是把一圈乘起来那么简单，因为周围格子互相之间有关联，不是独立事件。正确计算方法是先计算第一格不为雷概率；再假设第一格确定不为雷时，第二格不为雷概率；再假设第一二格都不为雷时第三格不为雷概率，以此类推把周围一圈格子及其本身的概率全部乘起来，技术上应该不难。有了破空概率算法后，让 AI 刷 IOE 也是个很有意义的事情。

4. 总结

随着计算机技术的发展，越来越多的作者将目光移向了计算机 AI 在扫雷中的应用。然而目前的研究大多不规范，主要体现在未声明细节及没有可验证性上。作者之间存在“重复造轮子”，以及开展研究前不查文献，重蹈覆辙的现象，鲜有创新性的研究。自动扫雷程序对扫雷学的发展具有重要意义，可以解决许多悬

而未决或有争议的问题。希望本文可以为后来研究者提供重要信息，做一个参考。

申明：笔者不从事自动扫雷算法研究，并与上述作者没有利益冲突，仅希望本文可以进一步深化相关研究，得出更有意义的结论。

参考文献

[1] Mathias.M. minesweeper solver[EB/OL]. http://www.wsdh.org/?minesweeper, 2008.02.02/2019.7.7.

[2] Vespa. 自动扫雷机[EB/OL]. http://www.kylen314.com/archives/762, 2014.01.07/2019.7.7.

[3] Kaye R . Minesweeper is NP-complete[J]. The Mathematical Intelligencer, 2000, 22(2):9-15.

[4] 龚秋源. 扫雷解局学[M/OL]. https://www.docin.com/p-2185963636.html, 2019: P41-42.

[5] Andrew Fowler. Minesweeper: A Statistical and Computational Analysis[D]. http://www.minesweeper.info/articles/MinesweeperStatisticalComputationalAnalysis.pdf, 2004.05.01/2018.12.26.

[6] **王嘉宁. 基于新判雷算法的猜雷次数计算[EB/OL].** http://www.saolei.net/BBS/Title.asp?Id=17163, 2019.3.11/2019.7.7.

[7] 王泽尧. python 自动扫雷程序实现思路简介和结果[EB/OL]. http://www.saolei.net/BBS/Title.asp?Id=17301, 2019.5.2/2019.7.7.

[8] 龚秋源. 扫雷解局学[M/OL]. https://www.docin.com/p-2185963636.html, 2019: P69-70.

[9] 王泽尧. 真·自动扫雷为你揭晓扫雷不为人知的秘密[EB/OL]. https://www.bilibili.com/video/av56120196, 2019.6.19/2019.7.7.

[10] 龚秋源. 扫雷解局学[M/OL]. https://www.docin.com/p-2185963636.html, 2019: P104-105.

[11] MinesweeperWiki. Board Cycles [EB/OL]. http://www.minesweeper.info/wiki/Board_Cycles ,2011.09.08/2018.12.29.

[12] Camargo, R. S. A possible explanation for the Board Cycle bug of Microsoft Minesweeper [J/OL]. http://www.minesweeper.info/articles/BoardCycleBug.pdf ,2006.09.23/2018.12.29.

[13] 龚秋源. 扫雷解局学[M/OL]. https://www.docin.com/p-2185963636.html, 2019: P64-66 & P133.

[14] 魔法小分队队长. 编程实现自动扫雷，我发现了惊天秘密！[EB/OL].https://www.bilibili.com/video/av8840844, 2017.02.26/2019.7.7.

[15] Undefind. 扫雷理论胜率 [EB/OL]. https://tieba.baidu.com/p/5103629698 ,2017.05.12/2018.12.29.

[16] 张少武. 猜雷与收尾无脑操作的简要教程——别把实力当运气[EB/OL]. https://zhuanlan.zhihu.com/p/35974785 ,2018.04.22/2018.12.23.