可满足性问题的算法设计与分析

（ 申请清华大学工学博士学位论文 ）

培养单位：清华大学计算机科学与技术系

专业方向：计算机应用

研 究 生：贺思敏

指导教师：张 钹 院士

答辩日期：1997 年 5 月 28 日

The Design and Analysis of Algorithms for Satisfiability Problem

Dissertation submitted to

Tsinghua University

in partial fulfillment of the requirement

for the degree of

Doctor of Engineering

by

Simin HE

Dissertation Supervisor: Prof. Bo ZHANG

May 28, 1997

可满足性问题的算法设计与分析

可满足性问题，即 SAT 问题，是计算机科学和人工智能的核心问题，以它为代表的 NP 完全问题在科学研究和实际应用中广泛存在，仅仅指出它们的难解性是不够的，更重要的是正面寻求解决方法，其中的关键是算法的设计与分析。

本文认为，局部搜索算法是对付 NP 完全问题最有希望的方法，实验分析将是研究局部搜索算法最主要的手段。本文针对局部搜索算法的灵活性、随机性和参数化等特点，深入探讨了相应的算法实验分析所涉及的算法性能度量、性能比较和参数调优等基本问题，提出以给定质量限下首次到达时间作为算法的性能度量，以秩和统计量为主，结合成功率、中位数、均值等统计量，从搜索过程到达的多个质量限观察和比较两算法的性能差距及其变化趋势，以全面地评价算法的性能，有效地发现改进算法性能的途径，并进一步提出应用 0.618 法进行算法单参数优化，从而给出了比较完整的局部搜索算法实验分析的概念和方法。文中以 SAT 问题的一个重要算法“GSAT+RandomWalk”的实验分析为例，给出了从数据收集、统计分析到参数调优完整过程的示范，提出了局部搜索算法设计的一个重要猜测“模拟升温”，展示了文中所提方法的可操作性和进行算法分析与设计的有效性。

本文对局部搜索初始点选择的一种新策略——划分策略进行了严格的理论分析。文中定义了划分类的均匀性偏序关系，定义了划分类的平均划分性能和最坏划分性能两个性能标准，证明了在任一个实例上，划分类越均匀，相应的初始点策略性能就越好，给出了作为最优划分策略的均分策略的性能上下界，分析了划分策略的实质，完整彻底地解决了对划分策略的评价问题，并对试验设计方法与初始点策略及局部搜索算法的关系进行了一定深度的讨论。

由于 NP 完全问题具有本质困难，在以局部搜索作为主要武器的同时，仍然要进一步探索新的求解思路。一种比较现实的思路是变换求解，

即通过某种变换，利用其他领域中比较有效的问题求解方法和策略实现间接求解原问题。最常用的变换求解是输入变换，特点是把问题输入变过去，把解答变回来，但中间求解过程是一黑箱，因此具有相当的局限性。本文提出了可读性变换的新思路，即把新问题形式下的方法通过一定方式用原问题形式下的概念和操作予以重新表述，从而形成原问题形式下可以理解的求解方法。在可读性变换的思想指导下，我们研究了用吴方法，即吴文俊消元法求解 SAT 问题的特点。通过设计合适的输入变换，建立了吴方法的基本操作与子句间有限制的归结操作的对应，从而证明了吴方法求解 SAT 问题时基本上是以特征列计算为核心的有限制的子句归结过程，使吴方法作为求解 SAT 问题的一种全新的方法具有了可读性。文中进行了全面的实验。可读性变换不仅比输入变换具有更高的效率，而且是算法设计的一种有效方法。

关键词：可满足性问题，局部搜索，算法实验分析，可读性变换，吴方法

The Design and Analysis of Algorithms for Satisfiability Problem

The propositional satisfiability problem, or SAT, is fundamental to computer science and artificial intelligence, and is representative of NP-complete problems, which widely exist in scientific research and practical applications. It is not enough to merely point out that NP-complete problems are intractable. What's more important is to explore ways of solving them, and the key to this is the design and analysis of algorithms.

Local search, in my opinion, is the most promising approach to solving NP-complete problems, and experimental analysis will be the main tool to design and analyze local search algorithms. In this dissertation, we present a systematic method to do such experimental analysis of local search algorithms. Local search algorithms are flexible, randomized and parameterized, and the corresponding experimental analysis must show how to measure and compare algorithm performance, and how to optimize algorithm parameters. We propose to measure algorithm performance by the first-success time under given quality bound, to compare algorithm performance by Wilcoxon two-sample rank-sum statistic combined with success rate, median and mean statistics, to observe the performance difference between two algorithms and its trend under various quality bounds during the whole search procedure in order to gain insight into the design of algorithms, and to apply 0.618 method to parameter optimization. We demonstrate this method by experimental analysis of "GSAT+RandomWalk", one popular local search algorithm for SAT, showing the elegance of the method.

We also give a theoretical analysis of a new approach to choosing initial points in local search, namely, the partition-based strategy. First, we define a partial order in partition types and two performance measures of the partition-

based initial point strategy, *i.e.*, the expected and the worst performance of a partition type. Second, we prove that on any instance, the more uniform the partition type, the better the performance of the corresponding initial point strategy. Third, we give the lower and upper bounds to the performance of the initial point strategy based on the even partition type, which is the best one of the kind. We further point out the essence of the partition-based initial point strategy and thus completely solve the problem of performance evaluation of this new initial point strategy for local search algorithms. We also discuss the relation of experimental design methods to initial point strategy and local search algorithms.

Since NP-complete problems are intrinsically difficult, it is necessary to explore new problem solving methods. A realistic way is the transformation problem solving, *i.e.*, by appropriate transform, we can make use of some relatively mature methods, techniques or strategies developed in other problem domains to solve the problem indirectly. The commonly used one is the input transform, but since its intermediate computing procedure is a black box from the viewpoint of the original problem, this approach has many limitations. In this dissertation, we propose a new approach called readable transform, which expresses an algorithm for a new problem with concepts and operations under the original problem representation and produces a new algorithm for solving the original problem. We apply readable transform approach to solving SAT by Wu's method. By establishing the correspondence between the primitive operation in Wu's method and clause resolution in SAT, it is shown that Wu's method, when used for solving SAT, is primarily a restricted clause resolution procedure. Readable transform will be an effective method for designing algorithms.

Key words: satisfiability problem, local search, experimental analysis of algorithms, readable transform, Wu's method

目　　录

第一章 绪论 **1**
§1 局部搜索算法简介 1
§2 人工智能界的局部搜索热 5
§3 求解 SAT 问题的局部搜索技术评述 10
§4 算法的实验分析 17
§5 论文内容简介 21
第二章 局部搜索算法的实验分析 **25**
§1 基本概念 25
§2 算法的性能度量 29
§3 数据收集和性能比较 37
§4 算法实验分析示例 41
§5 算法参数的优化 47
§6 总结 51
第三章 局部搜索多初始点选择的划分策略的性能分析 **67**
§1 简介 67
§2 概念和符号 68
§3 不同划分策略的性能比较 72
§4 划分策略的性能估计 78
§5 划分策略的实质 80
§6 总结和讨论 81
§7 附录 83
第四章 用吴方法求解可满足性问题 **87**
§1 研究背景 87
§2 吴方法简介 90
§3 用吴方法求解 SAT 问题 93
§4 实验及分析 105
§5 总结 113
参考文献 **115**
作者完成的论文 **124**

第一章 绪 论

§1. 局部搜索算法简介

局部搜索算法是本论文研究的一个重要内容。本文作者对局部搜索算法的兴趣最初是因为听说这种算法在 1 分钟内求得了百万皇后的无冲突布局。因此，我们通过皇后问题来介绍局部搜索算法的基本概念。

皇后问题一般是指 8 皇后问题，即要求在国际象棋 8×8 的棋盘上放置 8 个皇后，使任两个皇后不在同一行、同一列或同一对角线，否则按国际象棋的规则，皇后之间要发生冲突。图 1 是 8 个皇后的一种无冲突布局。由 8 皇后问题可以自然推广到 N 皇后问题。

			*				
					*		
							*
	*						
						*	
*							
		*					
				*			

图 1

皇后问题是大学计算机系算法设计和程序设计课的经典内容。一般用回溯算法求解，用递归方式实现。当时我们做这个题时，发现递归程序设计非常容易出错，而且比较费内存，有的同学能算出 12 个皇后的解就很高兴了。因此当我听说 1 分钟内能求解百万皇后的问题时，感到非常惊讶。

更令人惊讶的是，局部搜索算法求解皇后问题从思路到实现都极为简单。局部搜索算法是从 N 个皇后一个有冲突的完整布局出发，不断局部调整皇后的位置，使皇后之间总的冲突数减少；若减少到 0，则当时的布局就是所求的解。设π代表 N 个皇后的任一种布局，f(π)代表布局π中所有皇后之间的冲突总数，π(i, j)代表布局π中第 i 个皇后与第 j 个皇后交换位置后生成的新布局，则算法可描述如下：

```
1. 随机生成 N 个皇后的一个布局π;
2. repeat
3.       flag := 0;
4.       for i := 1 to (N – 1) do
5.            for j := (i + 1) to N do
6.                 if f(π(i, j)) < f(π)
7.                 begin
8.                      π := π(i, j);
9.                      flag := 1;
10.                end
11. until flag = 0;
12. if f(π) > 0
13.      goto 1;
```

上面这个过程中包含了经典局部搜索算法的所有概念。为了应用局部搜索算法，首先要把原问题变成一种优化形式，即在某一可行域中寻求某一目标函数的最优值的形式。例如，皇后问题中，我们把寻找无冲突布局的原问题，变成在所有可能布局组成的可行域中寻求目标函数即布局中所有皇后之间的冲突总数的最小值。其次，要在可行域中定义一种邻域结构，即什么样的两点称为相邻。例如，皇后问题中，如果两个布局只有两个皇后的位置不同则称为相邻。然后就可以在邻域中反复迭代，开始局部搜索了，这其中有三步。第一步要选定一个初始点，局部搜索将从这个点出发。上例中采用随机点作为初始点(语句 1)是常用的初始点策略。第二步要确定在邻域中如何移动，可以采用遇到好的邻点就

移动的见好就走方式，如上例(语句 2~11)，也可以采用向邻域中最好的点移动的最速下降方式。经典的局部搜索算法要求每一步移动都要对目标函数值有所改进，因此常被称为贪心局部搜索。第三步要确定如何对付局部极值点。局部极值点就是此点的邻域中没有比它好的点。有的问题中，局部极值点一定是全局极值点，如凸规划问题。有的问题中，局部极值点的质量足够好，已经满足需要，则也不必再找全局极值了。当然，如果局部极值不满足要求，最简单的方式是独立重复，即从一个新的初始点开始另一次局部搜索，寻找新的机会。上例就是采用这种方式(语句 12~13)。

从上面的介绍中可以感觉到局部搜索算法非常简单。局部搜索算法是六、七十年代求解组合优化问题时发展起来的，Lin 和 Kernighan 两人在求解旅行商问题和图划分问题时发展了经典局部搜索算法的各种技术[Lin, 1965; Kernighan & Lin, 1970; Lin & Kernighan, 1973]，[Papadimitriou & Steiglitz, 1982]是本文作者看到的第一本专门用一章内容介绍经典局部搜索算法的组合优化算法教材，书中对经典局部搜索算法的形成历史、主要技术、应用成果和一些理论结果做了非常好的介绍。本文作者认为，今天任何一本介绍组合算法的书，如果没有介绍局部搜索算法，则必定是有缺陷的，因为局部搜索算法具有足够多的优点，有资格作为算法设计的一种基本方法。

局部搜索算法的第一个优点是简单而且通用。优化是非常广的一种模型，几乎所有的问题均可以看成是某种形式的优化问题，即在某个可行域中寻求某一目标函数的最优值。只要你再进一步定义可行域中什么样的两个点算是相邻，就可以应用局部搜索算法求解了。算法从思路上讲就是瞎子爬山，拐杖能探到的点就是当前点的邻点。程序实现相当简单，比回溯算法的实现要容易得多。不论你碰到多么陌生的一个问题，你在想不出什么方法时，总可以拿局部搜索算法先试一试。如果你有了一个好方法，自然也应和局部搜索算法比一比。

局部搜索算法的第二个优点是有效。皇后问题自不必说，更有力的一个例子是求解旅行商问题。1990 年，Bentley [1990]在 VAX 8550 机器

上求解 10^6 个城市的随机实例，3.8 小时内得到的解仅比最优解差不到 3.5%。本文作者在选题前，对局部搜索算法的效率没有亲自体会，因此特地请两位同学在他们的问题上试一试局部搜索的效果。一位同学是利用语言学知识做汉字识别后处理[常新功, 1994]。汉字识别时，一般对句子中每个待识别字按相似程度给出 10 个候选字，利用语言学知识进行后处理是根据大规模语料库统计出的字与字的同现频率，从每个待识别字的 10 个候选中各找出一个字，组成同时出现可能性最大的句子。这个问题可以用动态规划求得最优解，但用局部搜索算法求解，不仅简单、速度快，而且找出的近似解的实际识别效果优于用动态规划找出的解。这是由于模型与真实世界总有差异，模型的最优解不一定是问题真正的最优解。另一个同学做并行任务调度及处理机分配时，以前用的列表调度算法一定条件下有一定性能保证，是经典的近似算法；但用局部搜索算法求解时，速度快，质量好，而且可以更真实地考虑问题的需要[林成江, 1995]。这两个实验很有特点，第一个问题上局部搜索的近似解战胜了动态规划的最优解，第二个问题上没有性能保证的局部搜索战胜了有性能保证的列表算法，这坚定了本文作者对局部搜索算法的信心。

局部搜索算法的第三个特点是代表了相当一大类问题现有算法的共同特征。常见的一些具有多项式算法的问题，如最小支撑树、最大流、最大匹配，还有排序问题，算法上均是一种局部搜索的思想，而且具有局部最优就是全局最优的好性质。连续优化问题由于不能象组合优化问题那样至少可以通过枚举来求解，因此所有算法均是局部搜索性质的算法。近年来非常流行的计算智能中的一些方法，如模拟退火算法[Kirpatrick *et al.*, 1983]、遗传算法[Goldberg, 1989]，均可以看作是在经典局部搜索算法基础上的一种变形。

对于 NP 完全问题，一方面由于不大可能在多项式时间内求得最优解，甚至不大可能在多项式时间内求得有一定性能保证的近似解，因此使用局部搜索算法不必有任何顾虑，因为没有保证比它好的算法。而且局部搜索算法既可以对近似算法求出的解做进一步的优化，又可以随机逼近最优解，因而比确定性的近似算法灵活得多。另一方面，在旅行商

等问题上局部搜索表现出非常好的性能。因此，局部搜索算法将是对付NP 完全问题最有前途的方法。

局部搜索算法诞生以后，在运筹学界求解优化问题的应用中一直很活跃，在算法的复杂性理论研究中也有许多成果[Yannakakis, 1990]，但是进入人工智能界却很晚，我们在下一节专门介绍这一过程。

§2. 人工智能界的局部搜索热

90 年代人工智能界的经典问题约束满足问题(包括可满足性问题)的算法研究出现了一个热点——局部搜索算法热。

这个热点的产生首先是因为约束满足问题是一类非常重要的问题，几乎所有的 NP 完全问题和人工智能界绝大多数应用问题均可以用约束满足问题模型表示。约束满足问题的一般形式是，给定一些变元，每个变元分别有多个可能的取值，变元的取值之间存在一些约束，询问满足所有约束的变元取值是否存在。皇后问题、图着色问题都是典型的约束满足问题。

自然，约束满足问题具有相当的难度。因此，约束满足问题的研究要成为热点，还必须同时有一个相对有效的有一定前途的求解方法。

约束满足问题在 70 年代的人工智能研究中第一次成为热点，一个重要的原因是 Waltz 在三维线框图的理解中应用一种“过滤”算法取得了较大成功，这种成功刺激了研究，最终发展形成了约束满足问题的局部一致性概念和算法，Waltz 算法成为其中一个特例[Freuder, 1978]。这项成果形式漂亮，在与回溯算法和后来的局部搜索算法相结合时有实用价值。但是 90 年代以前，在回溯算法的大框架下，约束满足问题的算法研究总的说来进展不大，没有吸引更多注意。

约束满足问题在 90 年代第二次成为热点，一个重要原因是局部搜索算法在求解约束满足问题时表现出很高的效率，使人们产生了新的希望。局部搜索算法与传统的回溯算法有很大的区别，但是局部搜索算法并不是一个新方法，它在六、七十年代形成后，八十年代又以模拟退火算法

和遗传算法等形式出现，可以说一直非常活跃，为什么一直等到进入九十年代才想到用局部搜索算法去求解约束满足这个古老的问题呢？

带着这个问题回头看 90 年代以前的局部搜索算法研究，可以发现所求解的问题均是优化形式，这种形式下，近似解是有意义的，可以接受的；而约束满足问题一般为判定形式，要求回答有解或无解，这时局部搜索找到的近似解没有意义，或者说，原则上局部搜索算法不能肯定地回答这种问题。因此人们比较自然地认为只有回溯算法才能求解约束满足问题。

既然如此，局部搜索算法是如何克服这种思维定势，成为约束满足问题求解的热点呢？

1990 年，Minton *et al.* [1990] 报告了他们取得的一个结果：用一种属于局部搜索的称为“冲突最小化”的启发式搜索算法，一百万个皇后的无冲突布局可以在 SPARCstation1 上用不到 4 分钟(优化后不到 1.5 分钟)的时间找到。由于传统的回溯算法在“合理的时间内”无法找到 97 个皇后的解[Stone & Stone, 1987]，Minton *et al.* 的结果是惊人的。

Minton *et al.* 在文章中反复强调，他们的工作是受 Johnston 和 Adorf 设计的一种叫做 GDS (Guarded Discrete Stochastic)的神经网络的意外成功所激发而进行的。GDS 成功地解决了哈勃太空望远镜观察任务的调度这一复杂问题，而在这个问题上传统的回溯算法和运筹学技术表现不良。为了搞清历史发展过程，我们追溯了可以找到的一篇引文[Adorf & Johnston, 1990]。

在[Adorf & Johnston, 1990]中，二位作者指出，约束满足问题的算法研究一直局限在回溯搜索的框架下探求各种改进策略以提高回溯搜索的效率，而用人工神经网络求解约束满足问题则是一条完全不同的路。他们发现，用二状态神经元模型的离散 Hopfield 网络表示一般的二元约束满足问题，如皇后问题，不仅直接了当，而且要比连续神经网络速度快；但是仍然存在一个问题：网络常常收敛到一个稳定状态，所对应的约束满足问题的变元没有合法取值。以皇后问题为例，这相当于有的皇后没有放在棋盘上。为解决这个问题，他们在主网上耦合了一个辅助网，称

为 guard 网，保证变元一定要取到合法值，也就是随时把跑到棋盘外的皇后放回到棋盘上。他们还对常用的网络随机演化算法做了重要修改，引入了贪心策略，即优先修改输入与输出最不一致的神经元的状态。经过这样的改进之后，网络不会稳定在皇后放在棋盘外的状态上了，但是由于整个网络的联接矩阵不再对称，网络不再保证一定收敛到稳态。这在理论上是个大问题，在实际应用中并不是个大问题，因为人们可以定时停止，重新开始。由此换来的好处是找到解的速度加快了。他们在皇后问题上首先取得了很大进展，1024 个皇后的无冲突布局在 16Mb 的 TI ExplorerII 工作站上用了不到 12 分钟就找到了。在一类随机约束满足问题实例和一类图的 3 着色问题实例上表现也很好。他们也发现在一类联接较稀疏的图的 3 着色问题实例上算法常陷于局部极值点而找不到解。但总的说来，结果非常吸引人。

用人工神经网络求解约束满足问题不论效率如何，方法的本质与传统的回溯搜索算法完全不同了，而与局部搜索算法求解约束满足问题的本质相同。因此我们认为这是走向局部搜索算法求解约束满足问题的第一步。Adorf 和 Johnston 的工作的重要性在于真正地提高了人工神经网络求解约束满足问题的效率，这是局部搜索算法打破回溯算法垄断的唯一武器和关键一步。Adorf 和 Johnston 设计的 GDS 网络有三个特点：一是网络模型保持了约束满足问题本质上的离散性，二是网络演化中保持了状态对应的变元取值的可行性，三是网络演化中采用了某种贪心策略。这三大特点使 GDS 与经典的离散、贪心局部搜索非常接近了。可惜的是，Adorf 和 Johnston 不仅没有尽快脱掉人工神经网络的外衣，反而想把约束满足问题的其他表示和技术均集成在 GDS 的框架下，而这个框架不仅比局部搜索算法复杂，而且有一个根本的限制，就是网络的空间复杂度太高，例如求解 N 个皇后的网络空间复杂性为 $O(N^2)$，因此无法求解更大规模的问题。

Minton *et al.* 则彻底抛弃了披在 GDS 上的神经网络的外衣，认为 GDS 成功的本质就是贪心局部搜索，并通过重复 GDS 所做的全部实验验证了这个结论。Minton *et al.* 完全走到了经典局部搜索这个更简单、更一般、

因而更易被人接受和应用的算法框架上，并由此获得了比 GDS 高得多的效率，百万皇后的成功求解就是最好的例证。Minton *et al.* 的工作使人真正注意到经典的局部搜索算法求解约束满足问题的巨大潜力，因而被人广泛引用。

Adorf和Johnston、Minton *et al.* 在皇后问题上的成功引起了Selman *et al.* 对局部搜索算法的注意。他们认为，这种成功可能是因为皇后问题是个容易的问题，而不是因为局部搜索是个有效的算法。这个看法不是没有道理的，因为皇后问题的确是一个非常特殊的约束满足问题，例如，至少有二种直接构造皇后问题的解的方法[Abramson & Yung, 1989; Bernhardsson, 1991]。因此，Selman *et al.* 估计，在可满足性这样的 NP 完全问题上，局部搜索算法可能会失败，即容易陷在只有几个子句不满足的局部极值点上。

可满足性问题(Satisfiability Problem, 简记为 SAT 问题)一般是指合取范式可满足性问题，即给定一些布尔变元，如 x,y,z，布尔变元之间的约束关系用子句的集合表示，如{x∨~y∨z, ~x∨y∨~z, …}，其中“~”表示逻辑“非”，“∨”表示逻辑“或”，询问是否存在使所有子句都为真的布尔变元 0,1 赋值。SAT 问题是第一个 NP 完全问题，也是约束满足问题的一种形式。

Selman *et al.* 设计了一种子句定长的SAT问题随机实例模型[Mitchell *et al.*, 1992]，并用这个模型产生的实例比较了针对 SAT 问题设计的一种经典回溯算法 DP 算法[Davis & Putnam, 1960; Davis *et al.*, 1962]和一种贪心局部搜索算法 GSAT 的实验性能[Selman *et al.*, 1992]。出乎他们的意料，在困难的可满足实例上，局部搜索算法比回溯算法找到解的速度快得多！换言之，局部搜索算法并不总陷在局部极值点中，而是常常可以找到全局最优点！

Selman *et al.* 的工作对于局部搜索算法热的形成和发展具有重要意义。首先，SAT 问题比一般的约束满足问题形式简单，而且是第一个 NP 完全问题，与人工智能的推理问题有非常直接的关系。因此，SAT 问题更为根本，具有独特的重要性，在 SAT 问题上证明局部搜索算法的高效

对于局部搜索算法成为研究热点具有决定性的作用。其次，Mitchell *et al.* 给出了一种生成难解随机实例的简单方法，这对深入研究和发展局部搜索算法以保持热点、避免昙花一现至关重要。用随机实例测试算法性能已成为算法研究的基本手段和方法，关于这一点，后面将专门论述。在 Selman, Mitchell *et al.* 1992 年的工作之后，有关 SAT 问题以及局部搜索算法和回溯算法的研究文章大量出现，标志着一个热点的形成。

回顾这一热点的形成过程可以发现，尽管原则上，局部搜索算法存在许多不足，例如，可能反复陷入局部极值点，对不可满足实例不能证明无解，对可满足实例不能保证找到解，这些方面回溯算法具有优势，但是，局部搜索算法的现实有效性战胜了回溯算法原则上的优越性。而发现和证实这一现实有效性并不是靠理论分析，完全是靠实验。耐人寻味的是，首先在实验上迈出这一步的不是研究局部搜索算法的人，自然也不是研究回溯算法的人，而是研究人工神经网络的人。人工神经网络不是一种有效的方法，因此最后的果实被经典局部搜索算法夺走了。但是，是人工神经网络的大胆探索首先发现了希望。

应当指出的是，用局部搜索算法求解约束满足问题还有一位代表人物顾钧（Gu, J.），他 1982 年从中国科技大学毕业后赴美留学。1990 年，Sosic 和顾钧[Sosic & Gu, 1990]非常详细地报告了用局部搜索算法求解皇后问题的算法实现和实验结果，在 NeXT 个人计算机(Motorola 68030 微处理器, 25Mhz)上求解 50 万个皇后平均时间为 10106 秒。1991 年，他们通过构造冲突较少的初始点和使用更快的计算机 IBM RS6000，用局部搜索算法可以在平均 55 秒内找到 300 万个皇后的解[Sosic & Gu, 1991]。1992 年后，顾钧发表了一系列用局部搜索算法求解 SAT 问题的文章[Gu, 1992, 1993, 1994]。Selman *et al.* 认为顾钧测试用的实例不够难，用回溯算法也可以快速求解[Selman *et al.*, 1992; Mitchell & Levesque, 1996]，而顾钧认为自己的算法即使在 Selman *et al.* 所说的难例上也比 GSAT 快[Gu, 1993, 1994]，二方面存在一些学术争议。本文作者认为，顾钧可能更早、更直接地认识到可以把约束满足问题由判定形式化为优化形式后用局部搜索算法求解，此外顾钧 1992 年回国讲学对于推动国内用局部搜索算法求解

SAT 问题的研究起了重要作用。在本文中我们较多讨论 Selman *et al.* 的工作则是因为 Selman *et al.* 公开了他们的源程序，我们可以进行重复实验，从而更准确地理解他们的工作和开展我们自己的研究。

§3. 求解 SAT 问题的局部搜索技术评述

局部搜索算法求解 SAT 问题形成热点后，为了进一步提高局部搜索算法的效率，针对 SAT 问题的特点，又发展了一些有效的策略和技术。此外，由于局部搜索算法求解 SAT 问题实质上是把 SAT 问题由判定形式化成优化形式来求解的，因此局部搜索算法求解优化问题的技术也可以用来求解 SAT 问题。局部搜索的各种技术是局部搜索算法的灵魂和生命所在，因此，我们以求解 SAT 问题为例，介绍一些有代表性的局部搜索技术，并对有代表性的观点和问题做一些评述。同时，我们也希望能使大家对这个领域研究的特点有所了解。

3.1. GSAT 算法

GSAT 是 Selman *et al.* 较早提出的一个求解 SAT 问题的算法[Selman *et al.*, 1992]。我们首先介绍 GSAT 并不是说它最有效，而是因为它比较简单，可以对局部搜索算法求解 SAT 问题的基本概念有较好的把握，其他一些技术多是以 GSAT 为基础的改进。

```
procedure GSAT
输入：子句集α，参数 MAX-FLIPS，MAX-TRIES
输出：满足α的一个变元真值指派(如果找到)
begin
    for i := 1 to MAX-TRIES
        T := 一个随机产生的变元真值指派
        for j := 1 to MAX-FLIPS
            if T 满足α then return T
            p :=变反后可以使α中 T 所满足的的子句数目得到最大增
```

```
                加的命题变元
            T := T 将其中 p 的真值变反
        end for
    end for
    return "没找到满足α的变元真值指派"
end
```

用局部搜索的语言讲，目标函数是当前变元赋值下不满足的子句个数，我们的目的是使之最小化；邻域用的是 1 邻域，即 Hamming 距离为 1(只有一个变元取值不同)的两个点(即变元的一个真值指派)称为邻点。这两个概念是局部搜索算法求解 SAT 问题时常用的。

从搜索方式上看，GSAT 主要有两个特点：一是最速下降，二是平移。其中平移比较值得多说几句。

所谓平移，就是点在移动前后函数值不变。经典局部搜索算法一般只允许下降移动，这样可以自然停止在局部极值点上。但在子句定长随机 3-SAT 实例上实验时，Selman *et al.* 发现，如果仅允许下降移动，则所到达的局部极值点的质量太差；而允许平移则可以走出局部极值点，发现更多的下降机会，因而有效地提高算法的性能。我们的实验也证实了这一点。在局部搜索算法的研究中，经常会发现一个看似平常的小技巧却大大改善了算法性能。从经典局部搜索算法入门的人走到这一点并不那么简单，因为这使一次局部搜索(GSAT 中的一次 try)的收敛问题变得复杂了，这就要引入一个参数 MAX-FIPS，进而带来一个如何确定合适的参数值的问题。算法的简洁性与算法的有效性常常产生矛盾，而最终优美屈服于效率，这是本文作者研究局部搜索算法最大的感受。

在 GSAT 的搜索过程中，平移往往占 90%以上，而平移的信息总的说来比较含糊，有些盲目。部分原因是 SAT 问题的目标函数用的是不满足子句个数，分辨率有限。提高平移方向性或目标性的一个途径是设计新的目标函数，另一个途径是对平移进行更细致的区分，确定哪些平移价值更大。此外，更大的邻域如 2 邻域也值得尝试。

3.2. 加权策略

加权策略是[Selman & Kautz, 1993], [Morris, 1993]同时提出的，目的是走出局部极值点。局部极值点相当于一个凹坑，加权策略的目的是把凹坑填平，从而有了向新区域发展的机会。具体方法是给每个子句赋一权重。目标函数定义为不满足子句的权重之和。子句权重初始化为 1，当到达局部极值点时，不满足子句权重增加 1。Selman *et al.* 在子句加权之后重新选一随机初始点开始下一次局部搜索，而 Morris 则在加权后从原先的局部极值点出发开始下一次局部搜索。就我们在一种保证有解的随机 3-SAT 实例上的初步实验，Morris 的策略要有效得多，原因似乎是因为局部极值点的不满足子句数目不大，权重变化后对目标函数整体影响有限，而随机初始点又往往导致不同的搜索区域，因而上次加权对这次局部搜索影响较小，而 Morris 的策略则可以有效地增大一次局部搜索的深度。

加权策略非常简单易用，而且对不同结构的问题适应性很好。Cha 和 Iwama[1995]用多种结构的随机实例测试了多种局部搜索策略，发现加权策略最好。

目前的加权策略在子句加权时每次加 1，虽嫌简单，但一些直观的改进并不有效。一个值得探索的问题是可否结合 SAT 问题特性和局部极值点附近结构更有目的性地增加子句权重以提高效率。

另一个改进策略是动态加权。现在的加权方式是在到达局部极值点时才加权，而梁东敏 *et al.*[1996]则在局部搜索每步移动中均随机选一个不满足子句进行加权。他们把这种动态随机加权策略与下文介绍的随机游动策略结合，在求解 SAT 问题的一些结构化实例中取得了非常好的结果[梁东敏 *et. al.*, 1996]。

3.3. 随机游动(random walk)策略

Selman *et al.* [1994]提出了一个对付子句定长随机 3-SAT 实例非常有效的策略，即给 GSAT 增加一个以一定概率发生的随机游动，具体地讲，就是以概率 p 从不满足子句中随机选一个变元变反，以概率 1-p 进行

GSAT。“GSAT+随机游动”与模拟退火算法的精神是一致的，即通过一定的上升移动设法走出局部极值点。但是“GSAT+随机游动”与一般的模拟退火有两个大的区别：其一是随机游动的概率在整个搜索过程中保持不变，与恒温的模拟退火有些相近；其二是随机游动的变元限制在不满足子句中的变元上。Selman *et al.* [1994]通过实验说明，当随机游动概率 p 取合适的值时，“GSAT+随机游动”比一般的模拟退火算法(包括几何降温方式和恒温方式)和随机游动变元不加限制的方式均好。可以认为“GSAT+随机游动”是针对 SAT 问题特点而设计的一种模拟退火算法。

本文作者对于“GSAT+随机游动”和模拟退火算法一直抱有怀疑。第一个原因是，经典局部搜索算法非常干净，下降移动，局部极值点停止，独立重复，没有什么参数要调节，而“GSAT+随机游动”则要引入一个参数 p，模拟退火算法则要引入更多的参数，给实际应用带来许多负担；第二个原因是，Ferreira & Zerovnik [1993]证明了，无论是渐近、还是有限步、还是并行实现下的性能，经典局部搜索算法均优于模拟退火算法；第三个原因是，我们在环形邻域搜索方式下引入随机游动，用定长随机 3-SAT 实例测试，从[Selman *et al.*, 1994]建议的 p=0.5~0.6 到 p=0 每隔 0.1，发现 p=0 最好，p 越大性能越差，这进一步加深了我们的怀疑，使我们在相当长的时间里没有接受这一策略。

直到后来我们得到了 Selman *et al.* 的源程序，并按 GSAT 实现了算法，才肯定了随机游动的确能大大提高 GSAT 的性能。我们又重新检查了原先用环形邻域搜索方式实现的算法，用一种单参数优化的方法找到了 p=0.037，这一点附近的性能优于 p=0 处的性能，这才最终使我们接受了随机游动策略。

上面的经历表明，引入参数是必要的，参数的最佳配置与算法细节密切相关，有效的参数寻优成为设计和分析算法的新问题。但这并不是说参数越多越好，也不是说现有的参数配置大框架就一定是合理的。这几方面问题将在后面的章节中深入讨论。

3.4. 大步(large-step)跳转策略

Yugami *et al.*[1994]提出了一个称为 EFLOP(Escaping From Local Optima by Propagation)的策略用来跳出局部极值点。具体方法是：从当前局部极值点处的不满足子句中选一个变元变反，这样会使一些原先已满足的子句变得不满足；然后从这些新变成不满足的子句中选一个与跳出局部极值点过程中已变反的变元不同的新变元变反；重复这一过程，直至没有符合条件的变元为止。这时一般已转移至一个与原先局部极值点有不止一步距离的新点上，从这个点上再重新启动局部搜索。文中用定长随机 3-SAT 实例测试表明 EFLOP 策略对 GSAT 性能的改善比单纯独立重复要大，而且似乎实例越难，改善越大。

我们对此策略的兴趣在于，EFLOP 与局部搜索算法求解其他优化问题时发展的一种通用策略“大步跳转”非常接近，可以把 EFLOP 看作是针对 SAT 问题而设计的大步跳转策略。

大步跳转策略是在求解旅行商问题中形成的，Johnson[1990]综合了 Muhlenbein 设计的遗传算法和 Martin *et al.*[1992]的大步马尔科夫链(large-step Markov chain)策略求解旅行商问题的成功经验，提出了迭代局部搜索(iterated local search)的策略，即在一般的小邻域下的局部搜索到达局部极值点后做一个大邻域的转移，然后开始新的局部搜索。Lourenco[1995]在作业调度问题上肯定了大步跳转策略的有效性。

Johnson[1990]认为迭代局部搜索(iterated local search)有单性繁殖的意思，有意推广为一般框架，似想取代遗传算法。近些年来，在遗传算法的大旗下，取得了一些好的实验结果[Johnson, 1990; Kolen & Pesch, 1994; Aarts, *et al.*, 1994]，这表明遗传算法值得研究。但这并不意味着应当固守遗传算法的框架。目前，遗传算法至少有种群规模、杂交率、变异率三个参数，从对遗传机制建模的角度讲，这些参数是必要的；但作为一种优化技术来用，则应当进行严格的对比实验以确定各种参数的作用，在保持效率的前提下尽可能简化模型。从局部搜索的观点看，遗传算法比较独特的地方是种群个体间的杂交，因为变异基本上是一种 1 邻域的移动，是局部搜索的基本操作。从目前遗传算法的实现中看，基本上采用

先从种群各个个体分别进行局部搜索到达各自的局部极值点，然后在局部极值点之间进行杂交，再接着分别进行局部搜索到达各自的局部极值点，然后进行淘汰与复制，再重复上面的杂交—局部搜索—淘汰与复制过程，这个框架称为遗传局部搜索(Genetic Local Search)[Kolen & Pesch, 1994; Aarts *et al.*, 1994]。如果两个个体间的杂交与两个个体独立做大距离转移没有实质区别的话，我们可以认为遗传算法、遗传局部搜索与“局部搜索+大步跳转”没有根本区别。如果进一步把大步跳转与模拟退火算法中的高温(相当于大的跳转概率)联系起来，则有可能对遗传算法和模拟退火算法达到一个简化后的统一认识。当然这是个大胆的猜想，还需要全面的实验分析。

目前看，大步跳转策略的效果依赖两点，首先是一个强有力的局部搜索算法，其次是一个适合问题特点的跳转方式。

3.5. 连续优化

上面介绍的是在 SAT 问题离散的可行解空间$\{0, 1\}^n$中进行局部搜索的一些策略。求解 SAT 问题的另一个思路是把 SAT 问题化成连续优化的问题，利用连续函数优化即非线性规划理论中的技术来求解原本是离散的问题。这对于认识连续问题和离散问题、连续优化方法与离散优化方法的本质联系，进而达到深刻的统一具有重要意义。

用连续优化方法求解 SAT 问题的一个代表性工作是[Gu, 1994]。文中把 n 个变元的 SAT 问题从离散的$\{0, 1\}^n$变成 $\mathbf{R}^n$中的无约束连续优化问题来求解。以其中代表性较强的 UniSAT7 模型为例，设所给 SAT 问题实例为二个子句的集合$\{x_1 \vee \sim x_2 \vee x_3, \sim x_1 \vee \sim x_2 \vee \sim x_3\}$，则相对应的函数为 $f_1(y_1,y_2,y_3) = (y_1-1)^{2p}(y_2+1)^{2p}(y_3-1)^{2p} + (y_1+1)^{2p}(y_2+1)^{2p}(y_3+1)^{2p}$，其中 p 为正整数；两种形式下解的对应为：$x_i=1$ 对应 $y_i=1$, $x_i=0$ 对应 $y_i=-1$，其他情况无对应。这样，子句集在某一点(x_1,x_2,x_3)上全部满足的充要条件是相应的(y_1,y_2,y_3)上 $f_1(y_1,y_2,y_3)=0$。为了保证理论上有一定的收敛速率，目标函数有时改为 $f(y_1,y_2,y_3) = f_1(y_1,y_2,y_3) + f_2(y_1,y_2,y_3)$，其中 $f_2(y_1,y_2,y_3) = \sum_{i=1}^{3}(y_i-1)^2(y_i+1)^2$，解的对应显然不变。文中证明，如果 $f(y_1,y_2,y_3)<1$，则可以找到 y_1^*, y_2^*, y_3^*

$\in \{-1, 1\}$，满足 $f(y_1^*,y_2^*,y_3^*) = 0$，从而找到原问题的解。这个定理不难，但也不显然，是个有意思的发现，当然在到达 $f(y_1,y_2,y_3)<1$ 之前也会陷入局部极值点。文中提到了大量经典连续优化方法，如牛顿法、拟牛顿法、共轭方向法等，以及一些方法的收敛性质估计。[Gu, 1994]的特点是从数学角度较全面地考察了用连续优化方法求解 SAT 问题的可能性。

另一个代表性的工作是[李未&黄文奇, 1994]。从形式上看，文中也是把 SAT 问题化成一个连续函数进行优化，但是从数学角度不太好理解他们采取的变换形式。仍以$\{x_1 \vee \sim x_2 \vee x_3, \sim x_1 \vee \sim x_2 \vee \sim x_3\}$为例，文中方法是求函数 $u(x_1,x_2,x_3) = (1-x_1)x_2(1-x_3) + x_1x_2x_3$ 在$[0, 1]^3$上的最小值，解的对应很直接。这是一个有约束的优化问题。从数学上讲，无约束优化问题更易处理一些。那末，作者为什么采取这种非常规的形式呢?

这是因为，文中作者认为，只有有物理含义的数学模型才会是有效的数学模型。上面形式上看不太合理的函数形式在物理上却有很好的模型，即表示一个带单位负电荷的质点在带正电荷的金属板形成的静电场中的电势能。文中从物理模型出发，推导出相应的数学形式，并进一步模拟质点在静电场中沿能量梯度方向下降的特点设计出相应的算法求最小值点。文中的能量函数是一个调和函数，因此能量最小点只可以在边界取得，这是个有意思的性质。

文中的思想是拟物拟人的思想。黄文奇[1989]指出：“此方法的精神在于，在某些困难的算法问题面前，当纯粹数学的思维显得似乎枯竭的时候，我们可以到自然界或人类社会中去寻求启发，而在最后设计算法时又不必拘泥于自然与社会现象的绝对真实性。至于如何作适当的修正或想象，则以有利于原始的算法问题的解决为准”。自 1979 年提出这一思想以来，黄文奇 *et al.* 在装填(Packing)问题[1979]、覆盖(Covering)问题[1989]、空间利用问题[1991]、方格填充问题[1993]、雷达群监视目标群问题[1995]等问题上应用这一思想设计了近似算法，发展了许多有用的策略。滕弘飞 *et al.*在求解旋转锥体空间中圆柱体群的布局优化问题[1993]、转动圆桌平衡摆盘问题[1994]时也采用了这一思想设计算法。黄文奇和金人超用拟物拟人的思想设计的算法在求解定长随机 3-SAT 实例

上取得了非常好的结果[1996]。

本文作者认为，拟物拟人的思想是一种重要的算法设计思想。近些年流行的模拟退火算法、遗传算法均是这种思想的产物。在各种算法的最初形成阶段，数学的作用似乎并不很大，一些基本原理、直观想象、大胆类比反而起着更大的作用。当然，在算法的进一步深入发展中，数学的作用是极其重要的，借助数学的严格性我们可以超越直观的局限性。至于用连续函数的优化方法求解 SAT 这样的离散问题，从思路上完全可以借鉴，但是，在离散空间中目标函数的计算可以利用局部搜索的局部性而达到很高的效率，在连续空间中目标函数的计算则似乎无法利用这一特点。从比较保守的角度，我们同意华罗庚和王元的观点：“离散的问题用离散的方法来处理为妥”[华罗庚, 1984]。当然，最终要看进一步的实验结果。

§4. 算法的实验分析

从上面的介绍中我们可以看到，在经典的局部搜索算法的基础上，从生物、物理、数学背景或直观认识出发已经发展形成了许多新的技术，每种技术本身也花样繁多，例如大量的参数，在提高经典局部搜索算法效率的同时，局部搜索算法本身也复杂化了。面对实际应用问题越来越大的复杂性，面对自身发展带来的复杂性，局部搜索算法的设计与分析必须建立相应的理论和方法。

目前的算法理论实际上包括两个部分，一个是计算复杂性理论，另一个是算法设计与分析理论。计算复杂性理论可以看作是算法设计与分析理论的元理论，主要是研究算法复杂性的下界问题[Cook, 1982]，即证明什么样的计算是不可能的，结论常常是否定性的，而不是正面的构造性的，这是元理论的特点[Brady, 1977]。因此，计算复杂性理论对于正面解决问题没有直接帮助。而算法的设计虽然是正面的构造性的，但基本上依靠个人的聪明才智和高度技巧，尽管发展了一些方法和策略，但远远谈不上什么理论。算法的分析同样需要高度的技巧性，而且是以渐近

复杂性为基础的，即使这样，也只能分析最坏情况或平均情况。在一个真实问题上，在现有的计算资源条件下，算法的表现如何，靠这种算法分析是不会获得太多信息的，这种分析的结果对于算法设计也给不出什么有用的启发。对局部搜索算法的各种理论分析，都没有能解释为什么局部搜索算法在现实中的表现这样好，更没有对如何设计局部搜索算法提出什么建设性的意见来[Papadimitriou & Steiglitz, 1982; Yannakakis, 1990]。理论获得的荣誉与它的实际贡献是不相称的。因此，局部搜索算法的设计与分析理论应当与以渐近复杂性为基础的经典算法设计与分析理论有所区别。最重要的原则是这种理论必须保持局部搜索算法的基本特点：现实有效性，这包括正面的构造性、现实有限计算资源下的可操作性等。这个理论要以实验为基础，但不是简单地做几个实验，而是强调有效地设计实验、有目的地采集数据、正确地分析数据，对算法的性能做出分析和判断，进而设计更好的算法。这个理论的一个合适名称应当是“实验算法学”，也常称为算法的实验分析，当然现在还正在发展之中。

一般地讲，算法的理论分析与算法的实验分析是算法分析必不可少的两个方面，互相补充对方的不足。如果一个算法的理论性能已经分析得很清楚了，那么实验分析往往主要起一个验证作用，但应当强调的是，即使这种情况下，实验分析仍是不可或缺的。如果一个算法的理论分析非常困难，难以给出有价值的信息，特别是局部搜索这类算法求解 NP 完全这类本质困难的问题时，实验分析的重要性就显现出来了。

从局部搜索算法的研究中可以看到实验算法学与经典的算法设计与分析理论(可以称为理论算法学)具有一些不同的特点。我们先介绍几个比较明显的特点。

4.1. 实例模型

传统的算法分析着重讨论算法在所有可能实例上的最坏或平均的渐近复杂性，而局部搜索算法的实验分析主要是考察算法在典型、具体的实例上的现实性能表现。其中，各种随机实例模型起着重要作用。

最常用的随机实例模型是均匀分布的实例模型，这样生成的实例一般没有明显的结构。例如，在旅行商问题研究中，最常用的是[0,1]×[0,1]的平面单位方块内均匀分布的点集所表示的城市实例模型；在 SAT 问题研究中，目前大家比较公认的是[Mitchell *et al.*, 1992]提出的子句定长 k-SAT 随机实例模型，生成方式是：从 n 个变元中随机无放回取 k 个变元，每个变元以 0.5 的概率取反，生成一个长度为 k 的子句；独立重复这一过程 m 次，即可生成一个 n 个变元、m 个子句的定长 k-SAT 随机实例。

有趣的是，在 SAT 问题研究中发现，随机实例有难易之分。[Mitchell *et al.*, 1992]中指出，有的随机实例模型很难产生难解实例，而定长 k-SAT 随机实例模型也只有当 m, n, k 间满足一定关系时才会生成多数算法均感难解的实例。例如 k=3 时，只有 m≈4.25n 时，生成的实例才比较难，n=250、m=1075 的 3-SAT 随机实例要比 n=1000、m=50000 的 3-SAT 随机实例难解得多，即仅仅规模大并不一定真正难。难例的生成与计算中的某种“相变”现象联系密切，Cheeseman *et al.*[1991]较早指出这个现象，Artificial Intelligence vol.81, no.1-2, 1996.3 有关于“相变与复杂性”的专辑。国内，白硕、刘涛、卜东波最早开展这方面的研究[Bai & Liu, 1994; Bai & Bu, 1996]，柳常青[1997]也对相变现象做了分析。本文没有过多介绍这方面工作，主要因为与算法设计的关系不是太直接。顺便指出的是，在求解连续优化问题的演化算法研究中也发现，常用的 10 个测试函数存在很大缺陷[Whitley *et al.*, 1996]。

随机实例也可以具有明显的结构。例如，旅行商问题研究中，Bentley [1990]曾构造了 10 种非均匀、不规则结构的随机实例模型来测试算法的健壮性(robustness)；SAT 问题研究中，Dechter 和 Rish[1994]、Kask 和 Dechter[1995]构造局部集聚的簇形(clustered)随机 3-SAT 实例来难倒常用的局部搜索算法；图划分问题研究中，Johnson *et al.* [1989]也用两种结构的随机图来比较经典局部搜索算法和模拟退火算法的性能。用多种结构的实例研究算法的性能可以发现哪些算法或算法的参数配置适合哪种结构的实例，从而在遇到真实实例时可以针对性地选择有效算法。

有一种观点认为随机实例不能反映真实问题的特点，主张用真实世

界的问题实例来研究算法。应当说，尽可能广泛地测试各种结构的实例是必要的，随机实例并不是唯一选择。但是随机实例模型具有独特的优点：首先，随机实例模型下可以用不多几个参数生成任意规模的实例，可以用于研究算法的渐近性能，而且便于同行重复实验与交流成果，这是真实世界问题实例所不具备的特点；其次，随机实例可以具有结构，可以按真实应用特点来设计相应结构的随机实例，并可以方便地研究结构与算法性能的关系。因此，随机实例模型是算法实验分析的主要手段，而真实世界实例可以通过建立公共库而成为另一种有价值的测试手段。

4.2. 实例规模

算法实验分析并不过分看重渐近复杂性，但并不意味着不重视算法的可扩展性。事实上，由于实际问题规模在不断增大，算法实验分析要保证现实有效性，实验用的实例必须具有相当的规模。

Johnson[1990]引述了有关旅行商问题真实背景的规模数据：电路板钻孔要求 17000，VLSI 工艺要求 1.2×10^6。文中指出，100 个城市上各种算法的相对性能优劣与更大规模时的情况有较大区别，而且 100 个城市的问题目前的完备算法可以在现实时间内保证找到最优解，因此只有远远大于 100 个城市时才有必要考虑启发式算法。

从这个角度看，Hopfield 网络求解旅行商问题所取得的“成功”至少在实验上是值得怀疑的。国际神经网络协会主席斯华龄[1988]介绍说，Hopfield 和 Tank 对分布于边长为 1 的方形区域内的 10 个城市的旅行商问题作了数字模拟计算；但是 Wilson 和 Pawley 不能重复 Hopfield-Tank 的实验结果；斯华龄改进了 Hopfield-Tank 的算法，对城市数 N=5,10,15,20,25 做了计算，效果很好。如此小规模的实例下得到的结论实在太不足以让人放心！许多人爱讲 10 个城市的解空间为 181440，100 个城市的解空间约为 4.67×10^{155}，因而认为 10 个城市能找到最优解很了不起，100 个城市则根本无法找到最优解，这都是对组合优化算法现状缺乏基本了解的结果。

4.3. 程序实现

程序实现在经典算法分析中的重要性主要表现在数据结构优劣可以影响算法的渐近复杂度，在算法实验分析中的重要性也许更大。因为实验分析关心现实有效性，一个渐近复杂性好的算法在真实应用范围内并不一定优于一个渐近复杂性差的算法；渐近复杂度相同的算法也会由于常数倍的差别而导致可用性上的根本区别；况且渐近复杂性分析非常难，影响算法性能的因素很多，优劣取决于综合效应，因而以运行时间衡量算法的整体性能表现有时是唯一可行的判别依据。因此，程序实现效率直接影响对算法的最终评价。好的程序实现不仅可以使我们对算法的实际可用性有准确估计，而且从实验角度讲，同样时间内可以测试更大规模的实例，从而使结论更可靠。[Bentley, 1990]是这方面工作的典范。

Hopcroft 在他的图灵奖演说中谈到，60 年代的算法研究非常令他不满意：杂志上发表的算法仅给出在一些样本实例上的执行时间，几年后又有人提出一个改进算法，声称在同样的实例上速度更快，但搞不清这是由于计算机更快了，还是程序实现更好了，还是算法本身有改进。因此，Hopcroft 力主在最坏实例情况下的渐近复杂性基础上建立算法的分析理论,使之与计算机和程序实现无关。从我们上面对算法实验分析几个特点的介绍，似乎算法分析又倒退回到 60 年代的水平了？本论文第二章发展了一套局部搜索算法的实验分析方法，从中不难得出对这个问题的回答。

§5. 论文内容简介

本文作者认为，局部搜索算法是对付 NP 完全问题最有希望的方法，实验分析将是研究局部搜索算法最主要的手段。本论文第二章针对局部搜索算法的灵活性、随机性和参数化等特点，深入探讨了相应的算法实验分析所涉及的算法性能度量、性能比较和参数调优等基本问题，提出以给定质量限下首次到达时间作为算法的性能度量，以秩和统计量为主，

结合成功率、中位数、均值等统计量，从搜索过程到达的多个质量限观察和比较两算法的性能差距及其变化趋势，以全面地评价算法的性能，有效地发现改进算法性能的途径，并进一步提出应用 0.618 法进行算法单参数优化，从而给出了比较完整的局部搜索算法实验分析的概念和方法。文中以一个算法的实验分析为例，给出了从数据收集、统计分析到参数调优完整过程的示范，提出了局部搜索算法设计的一个重要猜测“模拟升温”，展示了文中所提方法的可操作性和进行算法分析与设计的有效性。

本论文主张以实验分析作为研究算法、特别是局部搜索算法的主要手段，是因为理论分析受数学工具的限制，难度很大，以之作为算法分析的主要手段不太现实。但是我们并不是排斥理论分析。如果一个问题能够在理论上得到彻底解决，自然是非常好的一件事。事实上，即使在我们建立算法的实验分析方法的时候，也一直在寻求理论的支持。实验并非万能，也有它的局限性。本论文第三章对他人提出的一种局部搜索初始点选择的新策略——划分策略进行了严格的理论分析。文中定义了划分类的均匀性偏序关系，定义了划分类的平均划分性能和最坏划分性能两个性能标准，证明了在任一个实例上，划分类越均匀，相应的初始点策略性能就越好，给出了作为最优划分策略的均分策略的性能上下界，分析了划分策略的实质，完整彻底地解决了对划分策略的评价问题，并对试验设计方法与初始点策略及局部搜索算法的关系进行了一定深度的讨论。有趣的是，证明中大量使用了局部搜索的思想。

本论文认为局部搜索算法是对付 NP 完全问题最有前途的方法，但并不认为局部搜索算法可以包打天下，单纯依靠任何一个方法要想有效求解 NP 完全这种难度的问题是不现实的。在以局部搜索作为主要武器的同时，仍然要针对问题特点大力发展多种技术，探索新的求解思路。一种比较现实的思路是变换求解，即通过某种变换，利用其他领域中比较有效的问题求解方法和策略实现间接求解原问题。最常用的变换求解是输入变换，特点是把问题输入变过去，把解答变回来，中间求解过程是一黑箱。本论文第四章提出了可读性变换的新思路，即把新问题形式下的

方法通过一定方式用原问题形式下的概念和操作予以重新表述，从而利用新问题形式下的方法和策略形成原问题形式下的求解方法。在可读性变换的思想指导下，我们研究了用吴文俊消元法求解 SAT 问题的特点。通过设计合适的输入变换，建立了吴方法的基本操作与子句间有限制的归结操作的对应，从而证明了吴方法求解 SAT 问题时基本上是以特征列计算为核心的有限制的子句归结过程，使吴方法作为求解 SAT 问题的一种全新的方法具有了可读性，并由此获得了一些重要认识。文中进行了全面的实验。文中提出的可读性变换可以充分利用原问题的特性，可以有机结合、合理裁剪各种问题求解策略形成更有效的方法，可以帮助我们认识不同问题以及不同求解方法之间的内在联系，不仅比输入变换具有更高的效率，而且是算法设计的一种有效方法。

第二章　局部搜索算法的实验分析

本文作者的基本观点是：NP 完全问题不论多么困难，由于它在研究和应用领域中大量出现，因此必须正面寻求解决方法；局部搜索算法是对付 NP 完全问题最有前途的一种方法；实验分析将是研究局部搜索算法最主要的手段。基于这样一种认识，本文作者进入了这个研究领域，但首先面对的是算法实验分析的巨大复杂性：测试实例既要有一定规模，同时还必须具有相当的难度；算法中有多个参数，每个参数可能有多个取值；为了对付算法的随机性，需要对每一组参数值下的算法独立重复采样多次；由于运行时间成为评判算法性能的重要指标，程序实现必须足够有效；常常需要成百上千小时的计算机运行工作量，产生的数据量也是惊人的；……。由于算法实验分析，特别是针对局部搜索算法的实验分析，还没有形成清晰的概念和系统的方法，我们常感实验分析劳而无功，不仅还没有成为独立、有效的算法分析手段，而且有被自身复杂性淹没的危险。因此，澄清基本概念，发展基本方法，是我们进行算法实验分析首先要完成的工作。本章从分析局部搜索算法的特点入手，探讨了局部搜索算法的性能度量(比什么)、性能比较(怎么比)和参数优化(怎么调)等问题，初步给出了比较完整的方法。

§1. 基本概念

1.1. 算法与问题

本章所说的算法均指局部搜索算法。局部搜索算法实际上是指一类算法，包括纯随机采样法(即 Monte-Carlo 算法)，经典的局部搜索算法(即只进行能够改进质量的邻域移动，到达局部极值点后独立重复)，模拟退火算法[Kirpatrick *et al.*, 1983]，遗传算法[Goldberg, 1989]等。

局部搜索算法的第一个特点是灵活性。局部搜索算法是逐步改进一个完整解，而不是逐步扩展一个部分解，因此随时都可以给出当前最好

解作为近似解。同时，随时间的增加，局部搜索算法(至少通过独立重复)可以随机逼近全局最优解，这一点又比确定性的近似算法有优势。

局部搜索算法的第二个特点是随机性。目前大家常称局部搜索算法是不完备算法，以可满足性问题求解为例就是说，对可满足实例算法不能保证找到解，对不可满足实例则不能证明不可满足性。实际上，这均是由于局部搜索算法的随机性，使算法不论在证明实例可满足还是不可满足时都只能给出概率意义上的性能保证，从我们习惯的确定性算法角度来看，就成为常说的不完备性。对照素数判定的随机算法 Rabin 算法，不难理解上述解释，只是局部搜索算法没有 Rabin 算法那样高的效率而已。这同时也启发我们思考局部搜索算法何时才会有效。

局部搜索算法的第三个特点是参数化。例如，模拟退火算法有初始温度、降温速度、每一温度下的搜索次数等参数，遗传算法有种群规模、杂交概率、变异概率等参数，每一组参数值确定了一个具体算法。关于局部搜索算法的参数化，我们在本章第 5 节中仔细讨论。

本章所说的问题是指组合优化问题和更为一般的优化问题，但主要以可满足性问题(Satisfiability Problem, 以下简记为 SAT 问题)为例。SAT 问题可以有二种形式，一种是优化形式，即求使给定实例中不满足的子句数达到最小的解；另一种是判定形式，即询问给定实例是否存在使全部子句都满足的解。局部搜索算法求解优化形式的问题很自然，而且是通过求解优化形式的问题来回答判定形式的询问的。因此，对于局部搜索算法的设计与分析来说，两种形式的问题没有太大区别。

1.2. 算法的性能

算法的性能是指算法在求解问题时所需要的计算资源，主要指时间，与所给出的解的质量的关系。本文所说的算法的性能是指实验可观测到的算法的现实性能，不是指算法理论分析中的渐近性能。

由于算法性能涉及两个因素，不便直接观察与分析，因此比较自然的方法是从一个因素的角度观察另一个因素相应的变化。例如，随时间投资的不同，质量收益的变化有什么特点；或者随质量要求的提高，所

需时间的变化有什么趋势。这样既方便对一个算法性能的分析，又方便对不同算法性能的比较。至于应采取哪个观察角度，则应看问题的需要。

局部搜索算法性能比较的文献中一个常见的错误是没有在相同时间条件下比较解的质量，例如[Golden & Stewart, 1985; Lourenco, 1995]。产生这种错误的原因首先是因为没有正确理解算法性能的内涵，其次是因为文中的局部搜索算法有一个停止规则，因而造成了不等时。事实上，对于局部搜索算法来讲，做到等时间下比较并不困难。一方面，由于局部搜索算法的灵活性，不论时间多么短，总可以给出一个当前最好解。另一方面，由于局部搜索算法的随机性，算法自身一般难以判断当前最好解是否已达全局最优。因此，除非时间不允许而人为停止，算法本身应当永不停止；即使一次局部搜索有停止控制，也可以通过独立重复来继续运行。Johnson [1990]在旅行商问题研究中发现，相对简单的一种局部搜索算法 Lin-Kernighan 算法与比较复杂的模拟退火算法相比较，就各一次运行而言，前一个算法速度快，但解的质量差；若通过独立重复从而保证同样时间，则前一个算法给出的解的质量始终优于后一个算法。因此，对于局部搜索算法，不仅能够做到在任意相同的时间限度下比较解的质量，而且若不这样做，结论可能没有意义。

局部搜索算法性能的二因素中，时间易理解，质量则有时有歧义。我们这里所说的 T 时刻算法给出的解的质量 Q，是指到 T 时刻时算法所发现的最好解的质量。这也是绝大多数文献所公认的。但遗传算法研究中的重要人物 De Jong 引入了两种有些奇怪的质量定义，一种是衡量所谓算法进行性能(ongoing performance)的在线性能度量(on-line performance measure)，定义为 $\frac{1}{T}\sum_{1}^{T} f(t)$，其中 f(t)为第 t 次函数计算之值；另一种是衡量所谓算法收敛性能(convergence performance)的离线性能度量(off-line performance measure)，定义为 $\frac{1}{T}\sum_{1}^{T} f^{*}(t)$，其中 f*(t) = best{f(1), f(2),…, f(t)}[Goldberg, 1989]。[刘勇 *et al.*, 1995]沿用了此种性能度量。我们认为这是不对的，但由于没有得到原始文献，难以进行深入评判。此外，遗传算法研究中常称某一算法效率高，只是有些“早熟”。我们不赞成使用

“方言”，早熟现象的实质就是算法无法进一步改进其质量。我们认为算法研究必须采用纯客观的、统一的性能标准。

1.3. 面向每一个实例求解的原则

算法的实验分析主要是通过考察算法在各种测试实例上的性能表现而进行的。测试实例一般包括随机实例和具有一定真实应用背景的实例。为了全面评价算法性能，进一步设计更有效的算法，应当尽可能多地测试各种实例。但是有一些工作是比较算法在多种实例上的平均性能，如[Golden & Stewart, 1985]，我们对此不是太赞成。

首先，实际应用中，我们每次面对的一般是一个具体实例。从求解这一个实例的角度讲，一个算法在这个实例上表现好就是好算法，至于它在所有实例或典型实例上的平均性能好与坏是不重要的。算法实验分析最主要的目的是为了求解问题，而不是一定要把各个算法排出个一二三名。这其中有一个重要的观念转变，即我们不再期望一个算法包打天下，而是要针对具体问题的特点，从参数化的一族算法中选择一个尽可能适合求解该问题的算法。算法的实验分析就是要发展一整套调节参数、选择算法的技术，开辟一条求解困难的组合优化问题的新路，而不是重复经典算法理论分析中的最坏情况分析或平均情况分析。比较算法在多种实例上的平均实验性能在观念上是与算法平均复杂性分析一致的，因而具有与后者相同的局限性。

其次，局部搜索算法由于自身的随机性，在一个实例上性能的确定要比确定性算法复杂。即使要进行多种实例下算法的平均实验性能分析，也要首先解决一个实例下算法的性能分析问题。从分析方法上讲，多实例下算法平均性能分析与一个实例下随机算法的性能分析有相近之处，但是一个实例下的算法性能分析可以严格遵守统计分析方法的应用条件，例如任意样本量，样本独立同分布性，等等，而多实例下算法平均性能分析不易满足这些条件。

第三，我们认为，随机实例模型作为算法实验分析的主要手段，在同一组参数值下生成的具有一定规模的问题实例存在大的结构差异的可

能性不大。以定长随机 3-SAT 实例模型为例，我们用实例在可行解空间中任一点上满足的子句个数作为对实例结构的一种探测。设变元数为 n，子句数为 m，$\forall x \in \{0, 1\}^n$，随机实例在 x 上满足的子句个数服从二项分布 $\mathbf{B}(m, \frac{7}{8})$，其变异系数为 $\frac{1}{\sqrt{7m}}$，m 越大，变异系数越小。由二项分布与正态分布的关系也不难理解这一点。实验中也发现，一个算法在一个随机实例上多次独立重复运行产生的结果分布，与算法在同一组实例参数值下的多个随机实例上各一次运行所得的结果分布非常接近。SAT 问题其它随机实例模型下也发现类似现象[Cha & Iwama, 1995]。因此，一个随机实例的代表性可以认为足够强。事实上，Johnson *et al.*[1989, 1991]已经采用不同随机实例模型各生成一个随机实例来测试、比较各种算法在每一个实例上的性能，但没有说明理由。

综合上述各方面的考虑，本文主张算法的实验分析应当面向每一个实例的求解。本文的算法实验分析就是用一个实例的求解来说明我们的方法。这个方法原则上可以应用到任一个问题实例的求解上。当然针对不同问题的特点，还要进一步发展这一方法，但我们认为面向每一个实例求解的原则是必须坚持的。

§2. 算法的性能度量

2.1. 给定时间限下所达质量

在用局部搜索算法求解各种组合优化问题的实验研究中，观察算法性能最自然、最方便、也最常用的方法是以时间为自变量，观察算法在不同时间给出的解的质量的变化过程，也就是以给定时间限下所达质量作为算法的性能度量[Johnson *et al.*, 1989, 1991; Johnson, 1990; Bentley, 1990; Ulder *et al.*, 1991; Aarts *et al.*, 1994; Lourenco, 1995]。

大多数文章在报告算法性能时只给出一个时间限下各算法所得的解的质量，通常以比较费时的算法(如模拟退火算法)运行一次的时间作为公共时间限。严格地讲，仅给出一个时间限下所达质量并据此做出比较结论是不够全面的，因为在一个时间限下表现较差的算法并不一定在其他

时间限下表现也差。比如，有的算法在较短时间内给出的解比初始解的质量好很多，但时间长了却没有大的改进，我们称之为短期速降能力强，长期改进能力差，经典的局部搜索算法便是这种特点；有的算法则相反，短时间内改进不大，时间长了却质量领先，即短期速降能力差，长期改进能力强，模拟退火算法常有这个特点。这与运动员有爆发力和耐久力的不同素质是一个道理，两种能力各有各的用处。原则上观察时间应当是连续而无限长的，实际操作中当然总有一个最长时间限的限制，但应当使观察时限有一定的长度，而且要给出多个时间限下质量变化的全面报道。

那么，如果一个算法在所报告的所有时间限下质量均比另一个算法差，或更进一步，在观察开始后任一个时间点上质量均比另一个算法差，是否可以肯定这个算法就一无是处、没有任何价值呢？我们看一个例子：

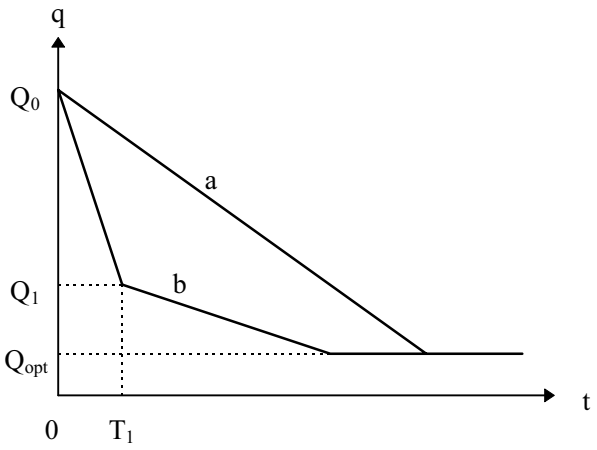

图 1

图 1 中，a、b 是两个算法，求解一个最小化问题，从 Q_0 出发，最后找到全局最优解质量为 Q_{opt}。这个过程中，按照任意给定时间限下所达质量来度量，可以说 b 算法性能始终优于 a 算法。

但仔细分析上图，在 $Q_0 \rightarrow Q_1$ 的过程中，b 算法比 a 算法快；但 $Q_1 \rightarrow Q_{opt}$ 的过程中，b 算法实际比 a 算法慢，只是由于在 $Q_0 \rightarrow Q_1$ 的过程中 b 算法积累的优势较大，尽管 $Q_1 \rightarrow Q_{opt}$ 的过程中，b 算法的优势在减小，但优势始终保持，因而从任一时刻观察，b 算法的性能始终优于 a 算法。如果没

有 $Q_0 \rightarrow Q_1$ 的优势积累，假设从 Q_1 质量的初始点出发，则 a 算法可能反而优于 b 算法！

当然，初始点并不一定可以任选。假如只能从 Q_0 出发，那么 b 算法无疑仍是正确的选择。这样看，上面的问题似乎又不是问题了。

但是，算法实验分析的目的并不只是从现有算法中挑一个最好的或排个一二三四，更重要的目的是设计比现有算法都好的新算法。设计新算法不能等待和依靠奇思异想，更为切实可行的道路是发展一套系统的方法去发现现有的最好算法是否还有不足，最差算法是否有闪光之处，在这个基础上扬长避短、优势互补来设计新算法。例如，上面 a、b 两算法可以是同一算法框架，假设是遗传算法，唯一区别是某一参数取值不同，假设是杂交概率取值不同，P_a=0.5，P_b=0.3。我们上面的讨论意味着，如果在 b 算法的后期将杂交概率提高到 0.5，则可能会有更好的性能。这样不仅设计了一个比原先算法都好的新算法，而且对原算法框架的不足有了更为深入的认识，从而可能突破原算法框架固有模式的束缚，创造新的机会。这就需要一个好的性能度量充当显微镜来细致分析算法性能。从这个意义上讲，给定时间限下所达质量这一性能度量即使采用任意多个时间限也不一定能发现问题，因而还不能满足算法实验分析的需要。

那么，是不是可以在多时间限下质量比较的基础上做一些改进呢？上面的讨论表明问题主要在于 b 算法初期的优势积累掩盖了 a 算法后期的努力追赶，因此不能光看 b 算法比 a 算法有优势，而且要看这种优势是在增大还是在减小，即要从多个时间限的质量比较中发现一些优势变化的趋势信息，也就是比较两算法的改善速度，即比较单位时间内质量改进量的大小，或相邻时限内质量改进量的大小。这样不就把前面积累的优势抵消了吗？以这个观点重新看图 1，T_1 之后，a、b 两算法给出的解的质量差距一直在减小，因此 T_1 之后 a 算法的改善速度快于 b 算法，a 算法的优点暴露出来了！

我们也一直以为这样就可以正确地分析算法性能了。但当我们总是发现明显差的算法却总表现出与好算法的质量越来越接近的趋势，才发觉上面的方法又错了。有二个原因：

第一个原因是，两个算法在相邻时限(或任意两个时限)内的质量改进分别是从不同质量的起点出发而完成的，而从不同质量的起点出发，完成同样的质量改进量具有不同的难度。这是由问题实例可行解空间的结构所决定的。因此，比较两个算法从不同质量的起点在单位时间或一段时间内完成的改进量是没有意义的，也就是说不可比。一般地，算法从好质量出发比从差质量出发完成同样质量改进量要难得多。这样，一个好算法在先到达好质量后，由于改进难度变大，因而改进速度降低；而一个差算法由于停滞在差质量，改进难度相对较小，改进速度反而显得相对比较高，因此从多个时限的质量比较看，会产生两算法质量差距反而缩小的现象，给人一种差算法正在追赶而且可能追上好算法的错觉！

第二个原因是，这种差算法追上好算法的现象是有一定必然性的。算法给出的解的质量是以实例目标函数的最小值为下限的，而局部搜索算法(至少可以通过独立重复)给出的解的质量可以随时间依概率收敛到这个最小值，即以此最小值作为所有局部搜索算法的解的质量的公共极限分布。这样，任意两个算法，即使是最好和最差的两个算法，在时间趋向无穷时，产生较大质量差异的概率趋于 0。因此当观察时限较长时，会发现所有算法的质量差不多了，也就是说差算法好象追上好算法了！

我们从多次失败的教训中得出的结论是：从时间角度，用给定时间限下所达质量只能孤立地比较各时间限下两算法质量高低，不能从多个时间限中分析差距变化趋势，因此不足以细致地分析算法的性能，必须寻找其它性能度量。但是，在给定时间限下所达质量这一性能度量的研究过程中提出的几个思想观点，例如不仅要观察一个时间限，而且要观察多个时间限；不仅要观察是否存在优势，而且要观察这种优势的变化趋势；不仅要在现有算法中找一个好算法，而且要在现有算法比较分析的基础上造一个更好、更新、甚至突破原有算法参数模式的算法，等等，均是非常有价值的，对于其它性能度量下算法的实验分析具有指导意义。

2.2. 给定质量限下首次到达时间

为了能正确而有效地进行算法性能分析，本文提出一个新的算法性

能度量：给定质量限下首次到达时间，即以质量为自变量，观察算法在不同质量要求下首次到达各质量限所需要的时间。

给定质量限下首次到达时间与给定时间限下所达质量具有很好的对称性，是从不同观察角度表示算法性能即时间与质量的关系，从图形上看只相差一个 90° 的坐标系旋转。在应用给定质量限下首次到达时间这一度量分析算法性能时，也要考察多个质量限，也要既看优势又看优势变化趋势，但是不会产生用给定时间限下所达质量进行同样的分析时可能产生的错误或错误导向：由于两算法性能差距及其变化趋势是在同样的质量起点或区间上比较，因而具有可比性；由于采用首次到达来计算时间，因而不会产生好算法等待差算法导致差距缩小的错觉。

为了体会给定质量限下首次到达时间这一性能度量在算法性能分析上的有效性，我们考察 Johnson *et al.* [1989] 在图划分问题上比较二种局部搜索算法性能时，用给定时间限下所达质量这一度量给出的数据，看看用新度量去分析会有什么新发现。

[Johnson *et al.*, 1989]是运用给定时间限下所达质量研究局部搜索算法性能的代表性工作，也是算法实验分析的经典工作。文中是以图划分问题进行实验的。给定图 G=(V, E)，其中 V 中含偶数个顶点，E 为边集；要求把顶点集 V 划分成二个相等大小的点集 V_1、V_2，使端点分跨在 V_1 和 V_2 中的边的数目最少。文中用此问题比较了模拟退火算法和另外二种局部搜索算法，一种是经典局部搜索算法，一种是 Kernighan-Lin(简记为 K-L)局部搜索算法，基本结论是：经典局部搜索算法性能最差；在均匀随机图上，如果时限较长，则模拟退火算法优于 K-L 算法，依据是表 1 给出的数据。其中，G_{500} 是一个 500 个顶点的均匀随机图，两点间有边存在的概率为 0.01；k 是算法独立重复次数；模拟退火算法运行一次所需时间为 K-L 算法运行一次时间的 100 倍，因此表中第 2 和第 4 列是相同时间限下的质量估计。文中的结论是：“Annealing clearly dominates K-L if running time is not taken into account, and still wins when running time *is* taken into account, although the margin of victory is much less impressive (but note that the margin increases as k increases).”

表 1：Comparison of Annealing and Kernighan-Lin on G_{500}

(Table I of [Johnson *et al.*, 1989])

k	Anneal (Best of *k*)	K-L (Best of *k*)	K-L (Best of 100*k*)
1	213.32	232.29	214.33
2	211.66	227.92	213.19
5	210.27	223.30	212.03
10	209.53	220.49	211.38
25	208.76	217.51	210.81
50	208.20	215.75	210.50
100	207.59	214.33	210.00

的确，从表 1 中看，模拟退火算法的优势即使时间限变到 100 倍也并没增大多少。但是，如果我们采用给定质量限下首次到达时间这一度量，则有表 2(时间单位为模拟退火算法一次运行时间)：

表 2

质量限	模拟退火算法 首次到达时间 t_1	K-L 算法 首次到达时间 t_2
213.32	$0 < t_1 \le 1$	$1 < t_2 < 2$
211.66	$1 < t_1 \le 2$	$5 < t_2 < 10$
210.27	$2 < t_1 \le 5$	$50 < t_2 < 100$
209.53	$5 < t_1 \le 10$	$100 < t_2$

这时我们会发现，模拟退火算法在上述质量区域内的改进能力远远优于 K-L 算法，时间需求相差可达 10 倍！

[Johnson *et al.*, 1989, 1991] 始终没有认识到用给定时间限下所达质量来分析算法性能时所存在的重大缺陷，更没有认识到运用合适的算法性能度量可以更有效地设计算法。在研究图着色问题时，他们也曾给出以质量为观察角度的图表，但仅仅是出于表示上的方便，而没有考虑以此来建立性能度量和分析方法。

本文是以 SAT 问题的求解为例来说明如何应用本文引入的给定质量限下首次到达时间这一度量来分析和设计局部搜索算法。目前，局部搜

索算法求解 SAT 问题的研究中，一般采用找到可满足实例的解的平均时间来比较算法优劣。从形式上看，这是以 0 个不满足子句为给定质量限来比较首次到达即找到可满足赋值的时间，但是这并不是我们所说的用给定质量限下首次到达时间进行算法分析的全部内涵，因此研究的广度和深度受到很大限制：

首先，现有方法一个非常大的限制是只能用可满足实例来研究局部搜索算法的性能，而构造比较大规模的既可满足又有相当难度的实例非常难，可以说还没什么方法，只能用一个较好的局部搜索算法在一定时间内试着从随机实例中找一个可满足的供研究用，自然规模不可能太大。

第二，即使有了可满足实例，现有方法还有一个不小的限制，即要求所有算法都找到解才好比较。但是实验时间总是有一定限度的，如果算法太差或实例太难，定时截断会造成大量数据不完整。比如，若找到解次数较多的算法找到解所需的平均时间比较长，如何判定优劣？若找到解的成功率普遍较低，如何产生可靠的结论？

第三，即使有了可满足实例，即使允许所有算法都找到解，现有的方法只是按平均成功时间给算法排个座次，而对算法整个求解过程的信息视而不见或不知如何利用，浪费惊人！

事实上，算法在找到解之前的全部变化过程的信息是极为丰富的。假设 a 算法与 b 算法是两个参数取值不同的算法，它们求解一个 SAT 问题实例的部分信息如表 3：

表 3

不满足 子句数	a 算法 首次到达时间	b 算法 首次到达时间
3	100	800
2	300	900
1	600	1000
0	900	1200

如果仅从找到解的时间来看，无疑 a 算法优于 b 算法。但仔细观察质量限从 3→2→1→0 的过程，发现 a 算法相对 b 算法的时间优势从

700→600→400→300，逐渐减小。这种优势存在却逐渐减小的趋势表明，在到达质量限 3 之前，a 算法要比 b 算法快，但是从 3 开始进一步下降的过程中，a 算法的能力比 b 算法弱，以前积累的优势因而逐渐减小。这表明，尽管 a 算法总体上比 b 算法快，但 a 算法在到达接近 0 的质量限时不如 b 算法性能好，因此应当借鉴 b 算法的参数取值特征来进一步提高 a 算法的性能。对 b 算法而言，在到达 3 之前与 a 算法的差距如何形成也是一个值得探讨的问题。这就是我们反复强调的要用多个界限，既看优势又看趋势以尽可能发现算法可以改进之处，进而设计好算法的思想。

如果 a 算法与 b 算法的求解过程部分信息如表 4：

表 4

不满足 子句数	a 算法 首次到达时间	b 算法 首次到达时间
3	100	600
2	300	1300
1	600	2600
0	5000	8000

这里同样是 a 算法先于 b 算法到达 0。仔细观察质量限从 3→2→1→0 的过程，会发现 a 比 b 的时间优势从 500→1000→2000→3000，即优势一直存在且越来越大。这时，实际上不必到达 0 就可以猜测 a 算法优于 b 算法的可能性极大。这意味着不必要求实例可满足，用不可满足实例照样可以进行算法研究。仔细想来，对于局部搜索算法而言，可满足实例与不可满足实例的区别主要在于算法可以达到的最好质量限(即最优解的质量)不同而已，这对算法的性能分析没什么区别。这样，分析与设计求解 SAT 问题的局部搜索算法就有了更广阔的天地！

以上我们分析了为什么常用的给定时间限下所达质量的性能度量不太适合用于算法性能分析，进而提出了一个新的算法性能度量，即给定质量限下首次到达时间，介绍了采用这一性能度量进行算法性能分析的基本思路，即从整个搜索过程所经过的各个质量限观察不同参数值的两个算法首次到达各质量限的时间的差距及其变化趋势，确定参数值对于

算法在不同质量起点下的改进能力的影响规律，进而设计新的参数配置以全面提高算法性能。但是局部搜索算法在任一给定质量限下的首次到达时间是一个随机变量，而不是上面示例中那样确定的一个值，因此，尽管我们已经明确应当比什么，但如何比仍是有待解决的问题，这是下一节的内容。

§3. 数据收集和性能比较

由于局部搜索算法的随机性，算法在求解任一个问题实例时，无论是给定时间限下所达质量，还是给定质量限下首次到达时间，都是随机变量，而随机变量比大小则要充分估计和控制随机性对结论的影响。因此，我们首先要收集算法在给定问题实例上多次独立重复求解的数据，然后设法利用数理统计理论提供的方法比较算法的性能。

3.1. 数据收集方法

本文采用的数据收集方法非常简单。在每一次对算法求解给定实例进行采样时，首先记录时间零点即采样开始时初始点的质量；然后每当算法首次改进到一个更好的质量时，记录此刻的时间和质量；本次采样由于到达预定时间限或预定质量限而结束时，记录此刻的时间和质量。

采用这种方法，即使采样时间极长，所需记录的数据量却非常有限。但从这些数据中，不仅可以自然获得算法首次到达任一质量限的时间，而且可以推算出采样过程中任一时刻算法所达到的质量。如果采用每隔一定时间间隔记录算法所达质量的方法，则不仅数据量会随采样时间增长而增长，而且不能精确推算出算法首次到达各质量限的时间。

利用这种方法获得的采样数据，我们可以对算法在整个搜索过程中任一阶段的性能进行全面而细致的分析。此外，可以通过对这些数据的再抽样来模拟和估计算法在独立重复和并行计算情况下的性能，一定程度上可以避免真实采样的巨大工作量。

3.2. 性能比较方法

当我们想应用数理统计理论中的方法来比较算法的性能时，面临的问题比较复杂。

首先，有多种观点的统计理论。除了传统的基于概率的频率解释的统计理论之外，还有贝叶斯统计理论和统计决策理论有可能用于我们的问题。我们在使用一个算法求解问题时，关心的是一次运行的结果，而用贝叶斯方法求出的解不需要频率解释，因此似乎很适合我们的需要。算法在应用中的结果往往涉及到一定的收益或损失，例如针对算法比赛中的计分规则，应选择一个什么样的算法参赛才能使得高分的可能性最大。因此算法比较涉及到决策问题，而统计决策理论可能给我们提供更多指导。当然各种统计理论都有优缺点。作为研究的第一步，我们仍然选用应用比较广泛的传统的统计理论来研究我们的问题。

其次，传统的统计理论有多种统计推断方法，各有各的模型假设。最常用的是总体分布为正态分布时基于抽样分布 t 分布的统计推断方法。此外有一些典型的非参数统计方法，如基于次序统计量和秩统计量的统计推断方法。本文对这些方法针对我们问题的应用做了研究。但至少还有一类方法可能也很适合我们的问题，即寿命数据统计分析方法。这主要有三个理由：一是可以把算法从开始搜索到找到给定质量要求的解而停止的时间看作一种寿命；二是我们实验过的几种局部搜索算法在求解定长随机 3-SAT 实例时首次到达给定质量限的时间分布与对数正态分布非常接近，而对数正态分布是一种典型的寿命分布；三是算法实验必然有一个停止时限，因而给定一个质量限，并非所有算法的所有独立重复运行均能在给定的时限内达到这一质量限，这样会产生大量定时截断数据，这是寿命数据的一大特点。因此利用寿命数据统计分析方法研究算法性能也是值得探索的方向。

我们在应用统计推断方法比较算法性能时，首先从尽可能多的角度分析样本。对单个算法样本，要计算均值、各种分位数、各种分散性度量、偏度系数和峰度系数。对两个算法样本，既使用基于 t 分布的方法分析，也使用基于秩统计量的方法分析；既进行显著性检验，也进行置信

区间估计；既进行成对比较，也进行非成对比较。每种方法如果有假设，则尽可能用合适的统计量估计假设的可靠性，如基于 t 分布的统计推断方法要求总体为正态分布，我们用样本的偏度和峰度系数对此做出检验。多个角度的互相对比可以对总体性质做出更全面的判断，对结论的可靠性有更准确的估计。

但是，同时进行多种统计推断会形成大量的数据和结论，如果没有一个主次之分，可能反而无法形成一个可用的认识，统计方法的应用就失去了意义。综合理论和实用两方面的特点，本文认为，算法性能比较应当以两样本秩和统计量为主要依据，理由如下[陈希孺&柴根象, 1993; Bickel, 1991]：

首先，两样本秩和统计量可操作性好，直观意义强。给定一个质量限，设 a 算法首次到达时间为随机变量 X，样本为 X_i, i=1,…, n_1；b 算法首次到达时间为随机变量 Y，样本为 Y_j, j=1,…, n_2。将 X_i 与 Y_j 混合起来从小到大排序，每个样本的序号称为它的秩；将 X 的各样本的秩加起来就是 X 样本的秩和，同理可得 Y 样本的秩和，依据任一个都可进行两样本同分布假设检验。从这个介绍中看，秩和统计量反映了单样本在合样本中的整体排列位置，偏小者表明样本总的说来排得较靠前，即首次到达时间偏小，因此直观上讲性能更好一些。此外，秩和统计量实质上是估计 **P**(X<Y)，即 X 与 Y 各随机取一个样本，哪一个小或优的机会大一些，这也是非常合理而且直观的。秩和统计量这种直观合理性可以帮助我们在进一步统计推断不可行时，做出比较正确的经验判断。

其次，秩统计量是非参数统计的一个主要工具，在理论上有很好的性质。对于两样本同分布假设的显著性检验，两样本秩和检验可以在总体为任何形式的分布下均能保证对第一类错误的限制，不必象两样本 t 检验那样原则上要求总体分布为正态分布。对于两样本位置参数的假设检验，两样本秩和检验仅在总体分布为正态分布时才比专门针对正态分布设计的 t 检验效率略有损失，但在非正态分布特别是厚尾分布下要比 t 检验在效率上有相当的优势。这意味着秩和检验要比 t 检验更可能检测出两样本位置参数存在的真实差异，即第二类错误的概率相对要小。这样，

我们使用秩和检验在理论上有充分的保证。特别是我们实验过的几个局部搜索算法求解定长随机 3-SAT 实例时，给定质量限下首次到达时间样本的偏度和峰度系数严重偏离正态分布的相应值，因此使用 t 检验非常令人不放心，必须进行对数变换后才可能使用 t 检验，这在其它问题上是否可行没有把握。秩和检验则没有这种负担，无论从实用还是从理论上看都值得作为一般方法推广。

第三，两样本秩和统计量要比常用的均值与中位数更好地反映两样本差异的整体情况。均值用到了全部样本，但是受个别样本的影响太大，由均值无法估计出一次抽样小于或大于均值的可能性有多大；而且在我们的问题中，常有定时截尾数据，既不能弃之不用，又无法或不愿假设总体分布形式，因而只能用一个估计值代替，而这个估计值的大小对均值影响极大，严重影响由均值所得结论的可靠性。中位数又正好相反，由于只用到一、二个样本次序统计量，因而不大受个别样本值的影响，截尾数据估计不会过分影响结论；但中位数缺乏对其他样本数据的反映，未必比其他分位数更合理多少。秩和统计量则把样本数据由小到大排列后以序号代替其真实值，相当于对数据做了一个标准化，这样就排除了个别值的过分影响；然后对序号求和，从而又很好地反映了全部数据的整体情况。

所以，本文比较两个算法的性能时，采用两样本秩和统计量为主要手段，既用它进行显著性检验，又用它来观察不同质量限下两算法性能差异的变化趋势，后者对分析算法更为重要。当然，我们不是只用秩和统计量观察，也要用其他统计量作为辅助观察手段。我们用两样本的中位数及其差，均值及其差来观察算法性能差异的绝对数量，观察这种差异从实用角度看是否很重要。由于存在截尾数据，因此我们还要观察成功率；当成功率太低时，说明截尾数据过多，由于截尾数据我们统一以二倍观察时限估计之，彼此不再区分，因而秩和统计量，以及其它统计量就不那么可靠了。下一节我们通过对一个算法的实验分析具体说明我们的方法。

§4. 算法实验分析示例

本节我们通过对一个算法的实验分析，示范性地说明前几节中我们提出的算法性能度量、数据收集、统计比较等方法，以及通过这一整套方法获得的一些深入的结果。本节的图表均列在本章末尾。

4.1. 实验说明

我们将要分析的算法是[Selman *et al.*, 1994] 提出的 “GSAT + 随机游动” 算法，主要分析随机游动概率 RandomWalkProb 对算法性能的影响。这一算法前面章节已介绍过，它是求解 SAT 问题的局部搜索算法研究领域中大家都很熟悉的算法，因此以它为分析对象更便于说明和理解我们的实验分析方法。我们在实现这个算法时，有二处小的改动：一是不采用 GSAT 中的最速下降策略，而采用下降优先于平移、对下降移动不再区分的策略；二是选择随机游动变元时，不是从不满足子句中的变元均匀随机选一个，而是先随机选一个不满足子句，再从中随机选一个变元。这样做的目的是提高效率，也有经验和个人喜好的原因，但没有本质区别。我们主要分析“GSAT + 随机游动” 算法不含定时停止独立重复时一次搜索进行下去所能达到的性能，这是算法的关键。

我们测试算法用的实例是通用的定长随机 3-SAT 实例，变元数 1000，子句数 4250，规模和难度相对较大。这个实例是可满足的，但我们的实验分析不依赖这个特点。

实验的硬件环境为 Pentium 100 Mhz, 16MB 内存的个人计算机，软件环境为 Linux 1.2.1 操作系统，全部程序用 C 语言编写。

实验采样方法是对每一个不同参数值(即 RandomWalkProb)确定的算法，从公共的 100 个随机初始点出发，相当于对算法采样 100 次，每次采样总时间为 300 万次变元翻转。

图 2 是随机游动概率 RandomWalkProb=0.600 时的数据文件示例。其中 BestUnsat 是指当前最少的不满足子句数，即算法当前最好解的质量；时间既用变元翻转次数记录，也以绝对时间(单位: 秒)来记录，以适应多

种分析需要；dGLS、sGLS 分别代表贪心局部搜索即 GSAT 阶段下降、平移累积次数，dRW、sRW、uRW 分别代表随机游动阶段下降、平移、上升移动累积次数，Thr 代表累积抖动次数，即接连的二次翻转是同一个变元的情况。这些信息是搜索过程顺便记录的，有时可以帮助分析。

我们的统计分析程序包括对单样本/两样本、给定时间限下所达质量分布/给定质量限下首次到达时间分布、按变元翻转次数计时/按秒计时、数据不变换/数据对数变换、基于秩和统计量分布/基于 t 分布、假设检验/置信区间估计等的分析，但主要以多个质量限下首次到达时间分析为主；时间以变元翻转次数计，便于别人重复；数据不进行变换，以便于方法推广。图 3 给出了 RandomWalkProb=0.500 和 RandomWalkProb=0.600 的两算法性能比较统计分析报告示例。

4.2. 随机游动概率对算法性能影响的实验分析

为了研究随机游动概率 RandomWalkProb 对算法“GSAT + 随机游动”性能的影响，我们测试了 RandomWalkProb = 0.000, 0.100, 0.200, 0.300, 0.400, 0.500, 0.600, 0.640, 0.700 时算法的性能，并对相邻两参数值下的算法性能进行了比较，实验分析报告的主要内容整理成表 5~表 12。

表中各项内容前面基本介绍过，另说明以下几点：(1) 在秩和统计量一栏中括号内给出的 BetterProb 是由秩和计算出的此算法与另一个算法在进行随机比较时获胜概率的估计，相当于对秩和统计量的一个归一化；(2) 当成功率 SuccessRate < 100%时，表明出现定时截尾数据，即在 300 万次变元翻转的观察时限内没有到达给定质量限。这时我们统一用 2 倍观察时限即 600 万次变元翻转去估计截尾数据的真实值，这样既与正好在时限 300 万时到达质量限的数据区别开，又可以继续进行其他统计量的比较。截尾数据对各个统计量有不同程度的影响，因此要结合成功率来理解其他统计量计算的结果；(3) 表中 DifferMedian 是指两样本随机比较差的中位数，一般不等于两样本中位数之差，这一点与均值不同；(4) “***”代表 10%水平上显著。

我们的实验分析方法第一个要点是观察多个质量限下两算法首次到

达时间的差距及其变化趋势。这一点是基础，但一旦论述清楚，操作上并不复杂。表中给出的质量限(即不满足子句数)有 0~10 每隔 1，10~20 每隔 2，20~100 每隔 10，100~500 每隔 20，几乎覆盖了从搜索开始到结束所经过的全部质量区域，从中可以发现比仅给出到达 0 的时间丰富得多的信息！

我们的实验分析方法第二个要点是借助数理统计方法来比较两算法性能差异。这一点从操作上讲，一方面要同时用多种统计量分析，另一方面又要根据数据特点对多种统计量有一个主次之分。表中给出了 4 种统计量：秩和统计量、成功率统计量、中位数统计量、均值统计量。在统计分析时，如果没有截尾数据，即成功率均为 100%，则秩和、中位数、均值都有效，但秩和整体感最好，应以秩和为主，中位数、均值为辅，互相对照、验证；如果开始出现截尾数据，即成功率<100%，由于均值受个别样本值影响太大，可靠性降低，而秩和、中位数不大受影响，继续有效，同时成功率的比较开始有意义；如果截尾数据超过一半，则中位数不再有效，秩和分辨率进一步下降，而成功率的重要性进一步上升；如果截尾数据过多，则说明有效数据太少，必须进一步增加观察时间才能形成可靠结论。

我们的实验分析方法的第三个要点是根据前二点，即多个质量限下两算法首次到达时间的差异及其变化趋势的统计分析结论，设计更新、更好的算法，这是我们算法实验分析的最终目的。而要达到这个目的，前二点的工作必须足够深入、细致。

下面我们结合具体算法的实验分析来示范说明我们的方法。首先分析 RandomWalkProb=0.000 与 RandomWalkProb=0.100 两算法的性能(参见表 5)。

由于定长随机 3-SAT 实例下，任一随机初始点的质量(即不满足子句数)约为子句总数(这里为 4250)的 1/8，因此搜索刚刚出发没几步就可以到达 500 这一质量限，我们从 500 开始分析。

质量限从 500→120 的过程中，从秩和统计量中可以清晰地看到 RandomWalkProb=0.000 相对 RandomWalkProb=0.100 的优势逐渐增大，

而且统计上达到显著。中位数、均值也证实了这一点，但从这两个统计量可以看出这种差异在实用中的重要性并不大，因为到达这些质量限都很快。

质量限从 120→16 的过程中，RandomWalkProb=0.000 的优势逐渐减小，从统计显著到不显著，最后反被 RandomWalkProb=0.100 赶上和超出，说明在这一质量区域 RandomWalkProb=0.100 的改进能力要比 RandomWalkProb=0.000 强。

质量限从 16→5 的过程中，RandomWalkProb=0.100 的优势继续增大，在统计上达到显著，在实用上也逐渐显著。由于开始出现截尾数据，因此均值的可靠性下降，成功率的重要性上升，但对秩和、中位数影响不大。注意到达 10 这个质量限时秩和有小的反弹现象，但不影响大的趋势。

质量限从 5→0 的过程中，截尾数据大量出现，因此中位数不再有效；由于截尾数据看作相等不再区分，因此秩和统计量的分辨能力下降，出现表面上差异缩小的趋势。但从成功率比较上可以看到差异仍很显著。总的说来，有效数据太少，统计量准确性大大下降。

从整个过程看，有两个极为明显的特点：一是在刚开始搜索时到达的质量较差的区域里，随机游动概率较小的算法(RandomWalkProb=0.000)要比随机游动概率较大的算法(RandomWalkProb=0.100)下降能力强；二是在质量限进一步提高时，即在质量要求较高的区域里，随机游动概率较大的算法改进能力要优于随机游动概率较小的算法。虽然两算法在观察时限内没有一次到达质量限 0，说明两个算法的性能都不好，但从我们的分析可以肯定 RandomWalkProb=0.100 的算法性能整体上优于 RandomWalkProb =0.000 的算法，这启示我们随机游动可能有用，不至于错过一个可能重要的发现。

用同样的分析方法可以发现表 6~表 10 中 RandomWalkProb=0.100, 0.200, 0.300, 0.400, 0.500, 0.600 相邻参数值下的两算法性能比较有同样的二大特点，直至表 11 和 12 情况才产生根本变化，我们下面对它们进行分析。

表 11 RandomWalkProb=0.600 与 RandomWalkProb=0.640 两算法性能

比较中，从质量限 500→80 的过程中看，RandomWalkProb 较小的算法表现出较大优势，且优势有逐渐增大的趋势，这一点与前面特点是一致的。从 80→0 的过程中，从秩和统计量看，RandomWalkProb 较大的算法开始追赶 RandomWalkProb 较小的算法，差距有缩小的迹象，但与前面特点不同的是，差距没有本质上的改变，即 RandomWalkProb 较小的算法始终保持了绝对的优势，其他统计量也证实了这一点。

而表 12 RandomWalkProb=0.640 与 RandomWalkProb=0.700 两算法的性能比较更具有说服力。整个过程中 RandomWalkProb 较大的算法始终没有表现出一丝追赶的迹象，而是劣势越来越大。这有力地说明了这样一个特点：增大 RandomWalkProb 并不总是可以提高算法在困难质量限下的改进能力，超过一定限度只会引起改进能力的全面下降！

这样，采用本文提出的算法实验分析方法，我们发现 RandomWalkProb 对 “GSAT + 随机游动” 算法的性能主要有以下影响：

(1) RandomWalkProb 越小，算法初期下降能力越强；

(2) 不超过某一限度，RandomWalkProb 越大，算法的后期改进能力越强；

(3) 超过一定限度，随机游动会引起算法性能的全面下降。

采用一定的 RandomWalkProb(=0.5~0.6)可以大大改善 GSAT 算法的性能，这是[Selman *et al.*, 1994]已经指出的，但是这一发现仅仅是通过算法求解可满足实例时找到可满足赋值(即达到质量限 0)的时间互相比较而得出的，文中没有提出系统的实验分析方法，因此也没有对这一发现深入分析。我们这里的目的不是为了验证[Selman *et al.*, 1994]中的结论，而是为了示范我们的实验分析方法如何把[Selman *et al.*, 1994]的初步发现引向深入。首先，我们的方法不要求实例可满足，而是通过对多个质量限下首次到达时间的趋势变化进行分析而发现 RandomWalkProb 的作用特点的。这意味着今后求解 SAT 问题的局部搜索算法的研究完全可以用任意实例来做测试，这就等于解决了生成大规模可满足的困难实例的难题。其次，我们的分析可适用于搜索全过程，因而发现了 RandomWalkProb 增大会引起算法早期改进能力的下降。虽然在定长随机 3-SAT 实例上，后期改进时间占大头，因而后期改进能力强更重要，可以弥补早期下降慢

的缺点，但是从全面提高算法性能上讲，这是重要的，特别是在一些优化问题如旅行商问题上，距最优解越近，改进越费时间，而真正的收益并不很大，因此提高算法早期下降能力就具有现实重要性了。

而最为重要的是我们从 RandomWalkProb 对“GSAT + 随机游动”算法性能影响的 3 个特点总结出这样一个猜测：

为了全面提高“GSAT + 随机游动”算法的性能，在搜索过程中应当把 RandomWalkProb 由小往大逐步提高(但不可超过一定限度)。

如果把 RandomWalkProb 与模拟退火算法中的温度联系起来，我们进一步猜测：

为了全面提高模拟退火算法的性能，在搜索过程中应当把温度由低往高逐步升高(但不可超过一定限度)。

当然这时严格讲已不再是“模拟退火”，而应称为“模拟升温”了；而且由于一时没找到物理背景，“模拟”也谈不上。当然名字不重要，重要的是内容。这个猜测来源于我们的实验分析，但由于临近毕业，无法马上进行实验验证。这个猜测的重要性在于它和我们熟悉的模拟退火算法在算法机制上正好相反！

模拟退火算法来源于物理背景，后来数学上也有了一些分析，但是真正说明模拟退火算法有用的却是实验，[Johnson *et al.*, 1989, 1991]是这方面工作的代表。实验证实，如果时间允许足够长，至少在某些问题上，模拟退火算法要比常用的局部搜索算法有效。但是，模拟退火算法的优点是否来源于退火机制呢？这种优点是否一定要以难以忍受的长时间为代价呢？对这些问题人们考虑极少。Johnson *et al.* [1989]发现初始温度太高没有用，太低性能也不好，但始终没考虑是否应当加温，即摆脱降温模式的束缚。而按我们的实验分析猜测，初始温度太高相当于 RandomWalkProb 太大，算法早期下降太慢，有很大浪费；初始温度太低，按降温模式，后续温度会更低，相当于 RandomWalkProb 过小，因而后期改进能力变差，性能自然也不好。Selman *et al.* [1994]也报告说，采用降温模式的模拟退火算法求解 SAT 问题性能极差，不如某一恒温方式好，但他们也没想到是否应当升温。而我们的实验分析猜测却容易解释为什

么降温方式不如恒温方式。康立山 *et al.* [1994]发现交替升温和降温的回火退火法找到了所有实验中最好的一个解，但由于多次重复退火过程，因此所用时间较长。这说明升温是有用的，而降温即使不完全抛弃，也至少应当有所控制。这也算是对我们猜测的一个佐证。

我们认为应当用研究 RandomWalkProb 对 “GSAT + 随机游动” 算法性能影响的方法来研究温度对于模拟退火算法性能的影响：首先对多个恒定温度下的搜索过程采样，然后用我们的实验分析方法进行分析，从中组合出一个最好的温度变化方式，看看是不是一种升温模式；最后对这种新模式进行实验，看看是否能产生预想的结果。进一步可以用这种方式研究遗传算法，特别是杂交操作对遗传算法性能的影响，我们猜测杂交操作的作用与模拟退火算法温度控制下的移动作用类似。

§5. 算法参数的优化

5.1. 关于算法的参数化

上一节的实验分析表明，在贪心局部搜索算法 GSAT 中引入一个参数 RandomWalkProb，当它取到一个合适的值时，可以比单纯的贪心局部搜索算法具有更好的性能。这是算法参数化带给我们的希望。但同时也应当看到，并不是所有参数值下算法的性能都可以提高，而寻找合适的参数值是要付出代价的。这是算法参数化带给我们的挑战。

参数化是当前局部搜索算法发展的重要特点。局部搜索算法一般有一些总的参数模式，如模拟退火算法、遗传算法，其中有一些参数，如 RandomWalkProb 这样的连续参数，翻转变元选择是否限于不满足子句这样的离散参数。参数化后的算法实质上是一族算法，每一组参数值确定一个算法。用参数化的算法求解问题时，不再是用固定参数值的一个算法对付所有的问题，而是针对每一个问题，设法寻找一组合适的算法参数值即设计一个新算法来求解。这样就把原优化问题的求解变成针对这个问题的算法设计，而算法设计实质上是算法参数空间中的一个优化问题。从形式上看，算法的参数化把一个优化问题的求解转化成另一个优

化问题的求解，有转嫁矛盾之嫌。这会带来实质性的进步吗？

这是个比较复杂的问题。理论上讲，NP 完全问题的求解不要指望奇迹的发生，无论是原先的优化问题，还是算法的参数优化问题，本质上都很困难。应用上讲，调参数不是个容易的事儿，目前有关模拟退火算法、遗传算法的成功报道，往往是最好的一组参数值下算法的结果，没有谈及寻找这组参数值所付出的巨大代价，因而具有一定片面性。但是我们讨论问题的一个根本出发点是无论 NP 完全问题多么困难，我们必须设法求解。现有的一些算法，不用说保证找到最优解的完备算法，就是经典的局部搜索算法，效率之低我们都是有切身体会的，难以让人看到希望。凭空设计一个有效的新算法是非常大的智力挑战，可以说无章可循。而算法参数化之后，参数调节意义上的算法设计要比传统意义上的算法设计相对简单多了，而且同样的时间内，通过参数调节得到的结果有可能比一个固定模式的算法给出的结果好。尽管这只是一种可能，但当我们身处绝境，发现算法参数化这根救命稻草时，首先应该看到的是稻草可能带给我们的希望，而不是它的脆弱性。碰碰运气总比坐以待毙好。这是我们面对困难问题时应当具有的现实态度，尽管本文作者的这种观念转变经历了相当长的过程。

从这一现实态度出发，重新研究算法参数化，发现还是可以有一些乐观的理由：

首先，算法参数空间的自由度比所要求解的问题实例的自由度少多了，例如模拟退火算法、遗传算法一般也就 4~5 个参数，却可以用于求解成百上千个变元规模的各种优化问题。参数空间中虽不是一个算法，却也不是所有算法。总的说来，算法参数空间的优化问题复杂性相对小一些，至少感觉上如此。

第二，虽然实际问题千差万别，但相应的算法参数的优化配置却有相对的稳定性。例如[Selman *et al.*, 1994]报告，用 “GSAT + 随机游动” 求解难解的定长随机 3-SAT 实例、规划问题、电路综合问题时，RandomWalkProb 最合适的值在 0.5~0.6 之间。这样只要我们对一个问题研究较透彻之后，所获得的参数优化配置可以对其他问题的求解有很好

的指导作用。

第三，即使不同问题之间算法参数优化配置区别较大，我们依然可以对问题按一些结构特征分类，相近结构的问题相应的算法参数优化配置比较接近。探索哪类问题适合哪类算法求解，是算法研究的一个重要方面。

因此，各种优化问题所对应的有效算法在算法参数空间中具有不同程度的相似性，这样我们可以针对一个问题或一类结构来寻找相应的算法参数优化配置，进而用于求解类似结构的其他问题。这就把一个真实问题的在线求解变成一个算法参数的离线优化，效率上的压力减小了。这是当前绝大部分局部搜索算法设计的根据。

上面的解释有一定道理，但毕竟算法参数空间中的优化不是一个容易的问题，即使离线研究也并不简单。因此，一方面，不要轻易引入参数；另一方面，提高参数调优的效率是算法实验分析与设计中一个迫切需要解决的问题。

算法参数调优实际上涉及三个问题：一个是算法的性能度量，即比什么；一个是算法性能的比较，即如何比；一个是如何调参数。前两个问题是算法参数调优的基础，本章前几节初步解决了这两个问题。本节主要讨论第三个问题，即如何调参数。

我们认为目前至少有两种方法有可能提高参数调节的效率：一种是 0.618 法，另一种是统计试验设计方法，特别是正交设计法。这二种方法的发展、应用中，我国科学工作者做出了重要贡献。这二种方法在理论上有一些好的性质，在实际应用中也经受了广泛的检验。我们对 0.618 法在局部搜索算法单参数优化中做了初步尝试，下面予以介绍。对正交设计法，我们没有来得及实验，但认为非常值得尝试。关于正交设计法，可参见《正交法和三次设计》(科学出版社, 1987)和项可风、吴启光所著《试验设计与数据分析》(上海科学技术出版社, 1989)。康立山 *et al.*[1994] 在模拟退火算法分析中应用了正交设计。

5.2. 单参数优化的 0.618 法

0.618 法是连续函数最优化的基本操作一维搜索的重要方法。华罗庚先生曾在我国大力推广 0.618 法即优选法，在生产实践中取得了大量成果[华罗庚, 1984]。

0.618 法最基本的应用是求闭区间[a, b]上单峰函数的极值。我们采用的是[陈宝林, 1989]第 305 页给出的算法。[华罗庚, 1984]中有对 0.618 法深入浅出的介绍。

我们对 0.618 法的兴趣在于，它在一元单峰函数优化中具有试验点最少的好性质；在多元单峰函数优化中也可以通过降维来使用；在多峰函数优化中，一方面可以通过空间的初步划分变成多个单峰问题后应用 0.618 法，另一方面找到一个峰从实用角度讲可能已经够用了。[华罗庚, 1984]对这些方面有讨论。此外，0.618 法在实际应用中经受了广泛的检验，这也是我们选择它的重要原因。

在我们所研究的局部搜索算法中，只有一个参数的算法有一些，如我们介绍过的“GSAT + 随机游动”算法主要是一个参数 RandomWalkProb。我们主要针对这个参数的优化试验了 0.618 法。

我们首先在“GSAT + 随机游动”算法上用 0.618 法对 RandomWalkProb 在[0, 1]区间寻优。试验点依次为 0.618→ 0.382→ 0.764→ 0.528→ 0.674→ 0.584→ 0.640→ 0.600，其中 0.600 最好。我们事先并不清楚整个区间[0, 1]上算法性能是否是单峰，但[Selman *et al.*, 1994]已报告 RandomWalkProb=0.5~0.6 较好，因此我们相当于是进行了一次验证。

我们随后在环形邻域搜索方式下用 0.618 法寻求 RandomWalkProb 的最佳值。环形邻域搜索方式是指翻转变元不是随机选择，而是按变元序号大小依次循环试探。这种方式是[Papadimitriou & Steiglitz, 1982]所推荐的，因而我们最初算法设计均采用环形方式搜索邻域。但我们发现按[Selman *et al.*, 1994]推荐的 RandomWalkProb=0.5~0.6，算法性能变得很差；以 0.1 的步长测试，发现均比 RandomWalkProb＝0 即不加随机游动策略性能差，而且 RandomWalkProb 越大，性能越差。因此非常怀疑随机游动策略是否有效。后来我们得到了 Selman *et al.* 的源程序，重复了

“GSAT + 随机游动”算法的实验，肯定了随机游动策略在 GSAT 算法下的作用，但是仍然无法在环形邻域搜索方式下发现随机游动策略的有效性。这样我们怀疑随机游动策略可能不是广泛有效的。但是不加随机游动策略，环形邻域搜索方式的性能优于 GSAT(表 13)，因此我们又不想放弃环形邻域搜索方式。于是我们决定试验 0.618 法，试验点依次为 0.618→0.382→ 0.236→ 0.146→ 0.090→ 0.056→ 0.034→ 0.021→ 0.043→0.030→0.037，发现随机游动概率 RandomWalkProb＝0.037 时算法性能最好，而且优于不加随机游动即 RandomWalkProb=0.000 时算法的性能(表 14)。

这个实验中算法的最佳参数值是非常微妙的，我们没想到它会距 0.5~0.6 那么远。这不仅肯定了随机游动策略的广泛有效，而且由于环形邻域搜索方式下 RandomWalkProb＝0.037 的性能远不如“GSAT + 随机游动”算法当 RandomWalkProb=0.600 的性能(表 15)，因而我们进一步否定了环形邻域搜索方式。通过这个实验，我们认为 0.618 法还是很有效的。通过更小间隔的穷举，不是不能发现 0.037 这个好值，但是进行这种实验往往令人望而却步。而有一个方法做后盾，我们实验的盲目性会大大减少，效率会大大提高，也就乐于进行必要的实验。而且也是通过这个实验，本文作者才终于承认算法参数化可能带来效率。

§6. 总结

本章针对局部搜索算法的基本特点，探讨了相应的算法实验分析所涉及的几个基本问题，初步给出了比较完整而且简单有效的方法。

局部搜索算法的第一个特点是灵活性，因而其性能表现具有多样性，这就要求相应的性能度量能全面、准确、细致地反映这一特点，这是有效设计新算法的基础。本文提出首先要从多个时间限或多个质量限的角度观察和比较算法的性能；其次不仅要观察两算法的性能差距，而且要观察这种差距的变化趋势；进而提出采用给定质量限下首次到达时间这一新的性能度量进行算法性能分析，解释了为什么常用的给定时间限下

所达质量这一性能度量不能满足算法性能分析的需要。

局部搜索算法的第二个特点是随机性，因此应当采用数理统计中的方法比较算法的性能。本文分析了为什么要选择秩和统计量作为比较的主要手段，提出了以秩和统计量为主，以成功率、中位数、均值统计量为辅的方法进行两算法性能差距及其变化趋势的分析。

本文以分析随机游动概率对算法“GSAT + 随机游动”性能的影响为例，示范了从数据收集到统计分析的整个过程，表明了本文提出的算法实验分析方法简单易行，而且所得出的分析结论对算法设计具有重要的启发。

局部搜索算法的第三个特点是参数化。本文分析并指出参数化在可能提高算法效率的同时，也带来了参数优化的新问题。因此一方面主张不轻易引入参数，另一方面则要尽力提高算法参数优化的效率。本文提出应用 0.618 法进行算法参数的优化，初步实验证实了 0.618 法在算法单参数优化中的效率。

本文提出的局部搜索算法实验分析方法比较完整，既十分注重良好的可操作性，使方法真正成为算法设计和问题求解的有力工具，同时也尽力寻求相关数学理论的支持，使方法具有进一步深入发展的潜力。本文提出的方法是我们大量实验的经验与教训的总结。

算法的实验分析历史不长，但它正在兴起，因为它是理论走向应用的桥梁。在当前崇尚“理论”的氛围中，真正认识到这一点的人并不多。不直接面对真实问题的人不大体会理论的局限性，不做实验的人也不大体会实验分析的艰辛。但我们相信，算法的实验分析最终会在求解真实问题的过程中表现出自己独特的价值。

```
c Time= Thu Dec 12 13:03:46 1996
c
c Sysname     = Linux
c Release     = 1.2.1
c Version     = #1 Mon Jun 24 13:19:55 CDT 1996
c Nodename    = saxophone
c Machine     = i586
c Domain      = (none)
c
c Program     = local1
c Function    = random neighbourhood search + greedy local search : downhill > sideways + random walk : in unsatisfied clause
c InstanceFile= instance1.v1000c4250.cnf
c       nVariable      = 1000
c       nClause        = 4250
c       nLiteral       = 12750
c InitialPointFile = initial.v1000n100
c       nVariable      = 1000
c       nPoint         = 100
c MaxFlip             = 3000000
c RandomWalkProb  = 0.600000
c RandomSeed        = 49384427
c
Program = local1
InstanceFile = instance1.v1000c4250.cnf
InitialPointFile = initial.v1000n100
MaxTry = 100
MaxFlip = 3000000
Sample = 1    nRecord = 379
nFlip=          0 Time=    0.00 BestUnsat=519    dGLS=0 sGLS=0 dRW=0 sRW=0 uRW=0    Thr=0
nFlip=          2 Time=    0.00 BestUnsat=518    dGLS=1 sGLS=0 dRW=0 sRW=0 uRW=1    Thr=0
nFlip=          5 Time=    0.00 BestUnsat=517    dGLS=2 sGLS=0 dRW=1 sRW=0 uRW=2    Thr=0
nFlip=          8 Time=    0.00 BestUnsat=513    dGLS=3 sGLS=0 dRW=2 sRW=0 uRW=3    Thr=0
nFlip=          9 Time=    0.00 BestUnsat=512    dGLS=4 sGLS=0 dRW=2 sRW=0 uRW=3    Thr=0
...
nFlip= 197487 Time=    6.62 BestUnsat=    4    dGLS=65152 sGLS=13755 dRW=14440 sRW=36397 uRW=67743    Thr=13587
nFlip= 258834 Time=    8.63 BestUnsat=    3    dGLS=84360 sGLS=19247 dRW=18840 sRW=48464 uRW=87923    Thr=18368
nFlip= 262183 Time=    8.73 BestUnsat=    2    dGLS=85381 sGLS=19587 dRW=19092 sRW=49077 uRW=89046    Thr=18713
nFlip= 366307 Time= 12.26 BestUnsat=    1    dGLS=116416 sGLS=30178 dRW=26828 sRW=70429 uRW=122456    Thr=27642
nFlip= 418664 Time= 14.14 BestUnsat=    0    dGLS=129690 sGLS=37869 dRW=30858 sRW=82775 uRW=137472    Thr=33480
nFlip= 418664 Time= 14.14 BestUnsat=    0    dGLS=129690 sGLS=37869 dRW=30858 sRW=82775 uRW=137472    Thr=33480
Sample = 2    nRecord = 378
nFlip=          0 Time=    0.00 BestUnsat=543    dGLS=0 sGLS=0 dRW=0 sRW=0 uRW=0    Thr=0
nFlip=          1 Time=    0.00 BestUnsat=542    dGLS=1 sGLS=0 dRW=0 sRW=0 uRW=0    Thr=0
nFlip=          2 Time=    0.00 BestUnsat=541    dGLS=1 sGLS=0 dRW=1 sRW=0 uRW=0    Thr=0
...
nFlip= 538640 Time= 18.06 BestUnsat=    2    dGLS=164263 sGLS=51607 dRW=38422 sRW=109387 uRW=174961    Thr=44416
nFlip=2750144 Time= 92.74 BestUnsat=    1    dGLS=835966 sGLS=265849 dRW=205473 sRW=545539 uRW=897317    Thr=234808
nFlip=3000000 Time=100.70 BestUnsat=    1    dGLS=908065 sGLS=293902 dRW=224079 sRW=598129 uRW=975825    Thr=258388
......
Sample = 100    nRecord = 369
nFlip=          0 Time=    0.00 BestUnsat=523    dGLS=0 sGLS=0 dRW=0 sRW=0 uRW=0    Thr=0
nFlip=          1 Time=    0.00 BestUnsat=522    dGLS=1 sGLS=0 dRW=0 sRW=0 uRW=0    Thr=0
nFlip=          2 Time=    0.00 BestUnsat=521    dGLS=1 sGLS=0 dRW=1 sRW=0 uRW=0    Thr=0
...
nFlip= 550528 Time= 17.79 BestUnsat=    2    dGLS=170096 sGLS=50603 dRW=40216 sRW=108895 uRW=180718    Thr=45377
nFlip= 619583 Time= 19.97 BestUnsat=    1    dGLS=190478 sGLS=57815 dRW=45330 sRW=123161 uRW=202799    Thr=51617
nFlip=3000000 Time= 96.07 BestUnsat=    1    dGLS=893385 sGLS=306979 dRW=220703 sRW=616285 uRW=962648    Thr=263510
```

图 2：数据文件示例

```
.....
c Program               = analyse7
c Function              = given two algorithm samples on same instance, given multiple quality bounds, output time (in flips) distribution
                          information, including {time , success-rate}×{unpaired , paired} ×{non-parametric Wilcoxon rank-sum , normal
                          distribution} ×{hypothesis testing , confidence interval estimation}. Note: (1) quality bound given can be outside
                          common quality interval; (2) if time used exceeds CommonTimeCutoff, we estimate the time to be
                          2*CommonTimeCutoff
c DataFile1             = data1.p0.500
c                       SearchProgram     = local1
c                       InstanceFile = instance1.v1000c4250.cnf
c                       InitialPointFile = initial.v1000n100
c                       MaxTry            = 100
c                       MaxFlip           = 3000000
c                       nSample           = 100
c                       AllQualityLow     = 4
c                       AllQualityHigh    = 465
c                       MillionFlipSec    = 31.797
c DataFile2             = data1.p0.600
c                       SearchProgram     = local1
c                       InstanceFile = instance1.v1000c4250.cnf
c                       InitialPointFile = initial.v1000n100
c                       MaxTry            = 100
c                       MaxFlip           = 3000000
c                       nSample           = 100
c                       AllQualityLow     = 2
c                       AllQualityHigh    = 465
c                       MillionFlipSec    = 32.067
c Report Quality Bound Interval = [ 0 , 100 ]
c ReportStep            = 1
c CommonTimeCutoff = 3000000
c Time 0 starts from time 0 of two datafiles
nSample_1 = 100 --- nSample_2 = 100 --- nPairedSample = 100
MaxFlip_1 = 3000000 --- MaxFlip_2 = 3000000 --- Common Max FLip = 3000000
Quality Interval_1 = [ 4 , 465 ] --- Quality Interval_2 = [ 2 , 465 ] --- Common Quality Interval = [ 4 , 465 ]
MillionFlipSeconds_1 = 31.797 --- MillionFlipSeconds_2 = 32.067
Report Quality Bound Interval = [ 0 , 100 ] --- ReportStep       = 1
CommonTimeCutoff = 3000000 --- Time 0 starts from time 0 of two datafiles

Quality Bound = 0 :
                        1. Time Distribution Difference CI and HT by non-parametric rank-sum and signed-rank-sum Statistics:
                            unpaired HT: rank-sum1 = 10662.50 , rank-sum2 =    9437.50 ,
                            unpaired CI: t1 - t2 = 0.00 ,          , 90 % CI = [ 0.00 , 0.00 ]
                            paired     HT: rank-sum-pos =    1150.00 , rank-sum-neg =      680.00 , *** , zero = 40
                            paired     CI: t1 - t2 = 77120.50 ,          , 90 % CI = [ 0.00 , 931774.50 ] , skewness = -0.13
                        2. Success Rate Difference CI and HT by U-Statistics:
                            basic      : p1 =      31.00 % , p2 =      43.00 % ( MedianTime :    t1 = 6000000.00 , t2 = 6000000.00 )
                            paired     : p1 - p2 = -12.00 %, *** , 90% CI = [ -22.98 % , -1.02 % ]
                            unpaired : p1 - p2 = -12.00 %, *** , 90% CI = [ -23.14 % , -0.86 % ]
                            HT             : ***
```

图 3：统计分析报告示例

3. Time Distribution Difference CI and HT by t Statistics :
unpaired basic : t1 = 4577781.31 , t2 = 4019459.01
unpaired CI(VE) : t1 - t2 = 558322.30 , *** , 90 % CI = [29658.19 , 1086986.41] , var1 = 4723545785222.15 , var2 = 5505001321057.04
unpaired CI(VNE): t1 - t2 = 558322.30 , *** , 90 % CI = [32216.76 , 1084427.84]
paired CI : t1 - t2 = 558322.30 , *** , 90 % CI = [36604.97 , 1080039.63]
4. time quantiles point estimation :
min_1 = 169746.00 T125_1 = 1038545.75 T250_1 = 2102954.00 median_1 = 6000000.00 T750_1 = 6000000.00 T875_1 = 6000000.00 max_1 = 6000000.00 c_v_1 = 0.47 std_var_1 = 2173371.99 ave_abs_dev_1 = 1962661.79 skewness_1 = -0.97 kurtosis_1 = 2.05
min_2 = 87605.00 T125_2 = 989636.75 T250_2 = 1489270.50 median_2 = 6000000.00 T750_2 = 6000000.00 T875_2 = 6000000.00 max_2 = 6000000.00 c_v_2 = 0.58 std_var_2 = 2346273.92 ave_abs_dev_2 = 2257816.73 skewness_2 = -0.43 kurtosis_2 = 1.33

Quality Bound = 1 :

1. Time Distribution Difference CI and HT by non-parametric rank-sum and signed-rank-sum Statistics:
unpaired HT: rank-sum1 = 12709.00 , rank-sum2 = 7391.00 , ***
unpaired CI: t1 - t2 = 846851.00 , *** , 90 % CI = [608129.00 , 1108453.00]
paired HT: rank-sum-pos = 4267.00 , rank-sum-neg = 683.00 , *** , zero = 1
paired CI: t1 - t2 = 1036071.25 , *** , 90 % CI = [709067.50 , 1694204.00] , skewness = 0.51
2. Success Rate Difference CI and HT by U-Statistics:
basic : p1 = 76.00 % , p2 = 97.00 % (MedianTime : t1 = 1597706.50 , t2 = 567602.00)
paired : p1 - p2 = -21.00 %, *** , 90% CI = [-28.46 % , -13.54 %]
unpaired : p1 - p2 = -21.00 %, *** , 90% CI = [-28.57 % , -13.43 %]
HT : ***
3. Time Distribution Difference CI and HT by t Statistics :
unpaired basic : t1 = 2405156.66 , t2 = 907255.69
unpaired CI(VE) : t1 - t2 = 1497900.97 , *** , 90 % CI = [1103219.15 , 1892582.79] , var1 = 4539841066434.71 , var2 = 1161126157912.34
unpaired CI(VNE): t1 - t2 = 1497900.97 , *** , 90 % CI = [1105129.28 , 1890672.66]
paired CI : t1 - t2 = 1497900.97 , *** , 90 % CI = [1111573.69 , 1884228.25]
4. time quantiles point estimation :
min_1 = 61243.00 T125_1 = 503748.12 T250_1 = 833149.00 median_1 = 1597706.50 T750_1 = 2883283.50 T875_1 = 6000000.00 max_1 = 6000000.00 c_v_1 = 0.89 std_var_1 = 2130690.28 ave_abs_dev_1 = 1766282.18 skewness_1 = 0.95 kurtosis_1 = 2.23
min_2 = 63470.00 T125_2 = 187090.00 T250_2 = 306771.50 median_2 = 567602.00 T750_2 = 1114042.50 T875_2 = 1596374.62 max_2 = 6000000.00 c_v_2 = 1.19 std_var_2 = 1077555.64 ave_abs_dev_2 = 656759.82 skewness_2 = 3.40 kurtosis_2 = 16.18

Quality Bound = 2 :

1. Time Distribution Difference CI and HT by non-parametric rank-sum and signed-rank-sum Statistics:
unpaired HT: rank-sum1 = 12704.00 , rank-sum2 = 7396.00 , ***
unpaired CI: t1 - t2 = 244156.50 , *** , 90 % CI = [179262.00 , 309556.00]
paired HT: rank-sum-pos = 4303.00 , rank-sum-neg = 747.00 , *** , zero = 0
paired CI: t1 - t2 = 322910.75 , *** , 90 % CI = [229337.00 , 450110.50] , skewness = 2.93
2. Success Rate Difference CI and HT by U-Statistics:
basic : p1 = 95.00 % , p2 = 100.00 % (MedianTime : t1 = 440865.50 , t2 = 226291.50)
paired : p1 - p2 = -5.00 %, *** , 90% CI = [-8.59 % , -1.41 %]
unpaired : p1 - p2 = -5.00 %, *** , 90% CI = [-8.59 % , -1.41 %]
HT : ***

.....

图 3：统计分析报告示例（续）

表 5：RandomWalkProb=0.000 --- RandomWalkProb=0.100 性能比较统计分析

Quality Bound	RankSum1 (BetterProb)	RankSum2 (BetterProb)	H.T. Sig.	Success Rate1	Success Rate2	H.T. Sig.	Median Time1	Median Time2	Differ Median	Mean Time1	Mean Time2	Mean Diff
0	10050.0 (0.500)	10050.0 (0.500)		0.0%	0.0%		6000000	6000000	0	6000000	6000000	0
1	10100.0 (0.495)	10000.0 (0.505)		0.0%	1.0%		6000000	6000000	0	6000000	5951906	48094
2	10547.5 (0.450)	9552.5 (0.550)		1.0%	11.0%	***	6000000	6000000	0	5943333	5469580	473753
3	10683.0 (0.437)	9417.0 (0.563)		10.0%	23.0%	***	6000000	6000000	0	5502414	4919866	582548
4	11581.0 (0.347)	8519.0 (0.653)	***	20.0%	53.0%	***	6000000	2322308	0	4950575	3327853	1622723
5	12031.0 (0.302)	8069.0 (0.698)	***	26.0%	72.0%	***	6000000	793534	3976310	4542375	2177512	2364863
6	11781.0 (0.327)	8319.0 (0.673)	***	45.0%	90.0%	***	6000000	432654	1988382	3467461	1160328	2307134
7	11390.0 (0.366)	8710.0 (0.634)	***	62.0%	95.0%	***	460021	174580	126266	2479942	644131	1835811
8	10935.0 (0.411)	9165.0 (0.589)	***	81.0%	98.0%	***	118886	82850	31034	1341408	361847	979561
9	10775.0 (0.427)	9325.0 (0.573)	***	87.0%	100.0%	***	60938	50187	12548	928852	129415	799437
10	10638.0 (0.441)	9462.0 (0.559)		90.0%	100.0%	***	43734	34906	6210	717257	99197	618060
12	10900.0 (0.415)	9200.0 (0.585)	***	98.0%	100.0%		23420	18249	4356	178806	26627	152179
14	10870.0 (0.418)	9230.0 (0.582)	***	99.0%	100.0%		14522	12194	2564	89320	16661	72660
16	10455.0 (0.460)	9645.0 (0.540)		100.0%	100.0%		9148	8280	678	13085	11251	1834
18	10232.0 (0.482)	9868.0 (0.518)		100.0%	100.0%		6796	6392	222	7989	7765	224
20	10023.0 (0.503)	10077.0 (0.497)		100.0%	100.0%		5076	4576	-18	5682	5364	318
30	10011.0 (0.504)	10089.0 (0.496)		100.0%	100.0%		1742	1769	-6	1849	1877	-28
40	9930.0 (0.512)	10170.0 (0.488)		100.0%	100.0%		998	1010	-9	1030	1036	-6
50	9288.0 (0.576)	10812.0 (0.424)	***	100.0%	100.0%		672	718	-31	706	729	-23
60	8941.5 (0.611)	11158.5 (0.389)	***	100.0%	100.0%		536	560	-28	540	569	-29
70	8564.0 (0.649)	11536.0 (0.351)	***	100.0%	100.0%		438	466	-28	445	472	-27
80	7802.5 (0.725)	12297.5 (0.275)	***	100.0%	100.0%		380	408	-33	379	413	-34
90	7138.5 (0.791)	12961.5 (0.209)	***	100.0%	100.0%		337	369	-34	337	370	-33
100	6483.5 (0.857)	13616.5 (0.143)	***	100.0%	100.0%		305	339	-35	307	341	-34
120	5737.0 (0.931)	14363.0 (0.069)	***	100.0%	100.0%		279	311	-33	278	311	-33
140	5958.5 (0.909)	14141.5 (0.091)	***	100.0%	100.0%		262	292	-30	260	289	-29
160	6155.0 (0.889)	13945.0 (0.111)	***	100.0%	100.0%		244	272	-26	244	270	-26
180	6367.5 (0.868)	13732.5 (0.132)	***	100.0%	100.0%		228	253	-23	228	251	-23
200	6661.5 (0.839)	13438.5 (0.161)	***	100.0%	100.0%		214	236	-21	212	233	-21
220	6960.0 (0.809)	13140.0 (0.191)	***	100.0%	100.0%		200	218	-18	198	216	-18
240	7108.0 (0.794)	12992.0 (0.206)	***	100.0%	100.0%		184	200	-16	183	199	-16
260	7433.0 (0.762)	12667.0 (0.238)	***	100.0%	100.0%		170	184	-14	169	183	-14
280	7601.5 (0.745)	12498.5 (0.255)	***	100.0%	100.0%		156	168	-13	155	168	-13
300	7680.5 (0.737)	12419.5 (0.263)	***	100.0%	100.0%		142	154	-12	141	153	-12
320	7955.5 (0.709)	12144.5 (0.291)	***	100.0%	100.0%		129	140	-10	127	138	-11
340	8165.0 (0.689)	11935.0 (0.311)	***	100.0%	100.0%		116	125	-9	115	124	-9
360	8362.5 (0.669)	11737.5 (0.331)	***	100.0%	100.0%		104	112	-8	102	110	-8
380	8640.0 (0.641)	11460.0 (0.359)	***	100.0%	100.0%		90	97	-6	89	96	-7
400	8901.0 (0.615)	11199.0 (0.385)	***	100.0%	100.0%		78	83	-5	77	82	-5
420	9130.0 (0.592)	10970.0 (0.408)	***	100.0%	100.0%		66	70	-4	64	68	-4
440	9247.5 (0.580)	10852.5 (0.420)	***	100.0%	100.0%		54	56	-4	52	56	-4
460	9390.5 (0.566)	10709.5 (0.434)		100.0%	100.0%		42	44	-3	40	43	-3
480	9572.0 (0.548)	10528.0 (0.452)		100.0%	100.0%		30	32	-2	28	31	-2
500	9766.0 (0.528)	10334.0 (0.472)		100.0%	100.0%		18	20	-1	17	18	-1

表 6：RandomWalkProb=0.100 --- RandomWalkProb=0.200 性能比较统计分析

Quality Bound	RankSum1 (BetterProb)	RankSum2 (BetterProb)	H.T. Sig.	Success Rate1	Success Rate2	H.T. Sig.	Median Time1	Median Time2	Differ Median	Mean Time1	Mean Time2	Mean Diff
0	10150.0 (0.490)	9950.0 (0.510)		0.0%	2.0%		6000000	6000000	0	6000000	5912758	87242
1	10349.5 (0.470)	9750.5 (0.530)		1.0%	7.0%	***	6000000	6000000	0	5951906	5673533	278372
2	10977.0 (0.407)	9123.0 (0.593)	***	11.0%	30.0%	***	6000000	6000000	0	5469580	4613372	856208
3	12192.5 (0.286)	7907.5 (0.714)	***	23.0%	65.0%	***	6000000	2046558	3258708	4919866	2879243	2040623
4	11984.0 (0.307)	8116.0 (0.693)	***	53.0%	82.0%	***	2322308	577312	859808	3327853	1672231	1655621
5	11828.0 (0.322)	8272.0 (0.678)	***	72.0%	92.0%	***	793534	310886	358774	2177512	907746	1269767
6	11794.0 (0.326)	8306.0 (0.674)	***	90.0%	99.0%	***	432654	162153	164483	1160328	413541	746787
7	11749.0 (0.330)	8351.0 (0.670)	***	95.0%	100.0%	***	174580	82016	69541	644131	231440	412691
8	11599.0 (0.345)	8501.0 (0.655)	***	98.0%	100.0%		82850	52300	30880	361847	98327	263519
9	11110.0 (0.394)	8990.0 (0.606)	***	100.0%	100.0%		50187	37576	11906	129415	59250	70165
10	11411.0 (0.364)	8689.0 (0.636)	***	100.0%	100.0%		34906	25538	9172	99197	39191	60006
12	10965.5 (0.408)	9134.5 (0.592)	***	100.0%	100.0%		18249	15790	3050	26627	20885	5742
14	10552.0 (0.450)	9548.0 (0.550)		100.0%	100.0%		12194	11692	1036	16661	12797	3863
16	10420.0 (0.463)	9680.0 (0.537)		100.0%	100.0%		8280	7566	530	11251	8627	2624
18	10367.5 (0.468)	9732.5 (0.532)		100.0%	100.0%		6392	5975	332	7765	6376	1390
20	10292.5 (0.476)	9807.5 (0.524)		100.0%	100.0%		4576	4411	166	5364	4952	413
30	9884.0 (0.517)	10216.0 (0.483)		100.0%	100.0%		1769	1748	-28	1877	1897	-21
40	8894.5 (0.616)	11205.5 (0.384)	***	100.0%	100.0%		1010	1084	-82	1036	1138	-102
50	8657.0 (0.639)	11443.0 (0.361)	***	100.0%	100.0%		718	766	-55	729	796	-67
60	8070.0 (0.698)	12030.0 (0.302)	***	100.0%	100.0%		560	604	-52	569	624	-55
70	7502.0 (0.755)	12598.0 (0.245)	***	100.0%	100.0%		466	516	-50	472	524	-52
80	7282.0 (0.777)	12818.0 (0.223)	***	100.0%	100.0%		408	454	-45	413	458	-45
90	6837.5 (0.821)	13262.5 (0.179)	***	100.0%	100.0%		369	410	-41	370	412	-42
100	6350.5 (0.870)	13749.5 (0.130)	***	100.0%	100.0%		339	382	-39	341	381	-39
120	6018.5 (0.903)	14081.5 (0.097)	***	100.0%	100.0%		311	353	-39	311	350	-39
140	6148.5 (0.890)	13951.5 (0.110)	***	100.0%	100.0%		292	324	-34	289	324	-34
160	6468.0 (0.858)	13632.0 (0.142)	***	100.0%	100.0%		272	300	-29	270	299	-29
180	6659.5 (0.839)	13440.5 (0.161)	***	100.0%	100.0%		253	278	-26	251	277	-26
200	6812.0 (0.824)	13288.0 (0.176)	***	100.0%	100.0%		236	258	-24	233	257	-24
220	7111.5 (0.794)	12988.5 (0.206)	***	100.0%	100.0%		218	238	-20	216	237	-21
240	7390.0 (0.766)	12710.0 (0.234)	***	100.0%	100.0%		200	216	-18	199	217	-18
260	7702.5 (0.735)	12397.5 (0.265)	***	100.0%	100.0%		184	198	-15	183	198	-15
280	7998.0 (0.705)	12102.0 (0.295)	***	100.0%	100.0%		168	182	-12	168	181	-13
300	8176.0 (0.687)	11924.0 (0.313)	***	100.0%	100.0%		154	165	-11	153	164	-11
320	8376.0 (0.667)	11724.0 (0.333)	***	100.0%	100.0%		140	148	-10	138	148	-10
340	8623.0 (0.643)	11477.0 (0.357)	***	100.0%	100.0%		125	132	-8	124	132	-8
360	8887.0 (0.616)	11213.0 (0.384)	***	100.0%	100.0%		112	117	-6	110	116	-6
380	9064.5 (0.599)	11035.5 (0.401)	***	100.0%	100.0%		97	102	-5	96	101	-5
400	9231.5 (0.582)	10868.5 (0.418)	***	100.0%	100.0%		83	87	-4	82	86	-4
420	9313.0 (0.574)	10787.0 (0.426)	***	100.0%	100.0%		70	73	-4	68	72	-4
440	9582.0 (0.547)	10518.0 (0.453)		100.0%	100.0%		56	59	-2	56	58	-2
460	9748.0 (0.530)	10352.0 (0.470)		100.0%	100.0%		44	45	-1	43	44	-1
480	9844.5 (0.521)	10255.5 (0.479)		100.0%	100.0%		32	33	-1	31	32	-1
500	9853.5 (0.520)	10246.5 (0.480)		100.0%	100.0%		20	19	-1	18	19	-1

表 7：RandomWalkProb=0.200 --- RandomWalkProb=0.300 性能比较统计分析

Quality Bound	RankSum1 (BetterProb)	RankSum2 (BetterProb)	H.T. Sig.	Success Rate1	Success Rate2	H.T. Sig.	Median Time1	Median Time2	Differ Median	Mean Time1	Mean Time2	Mean Diff
0	10152.0 (0.490)	9948.0 (0.510)		2.0%	4.0%		6000000	6000000	0	5912758	5801660	111098
1	10592.0 (0.446)	9508.0 (0.554)		7.0%	18.0%	***	6000000	6000000	0	5673533	5205951	467583
2	11162.0 (0.389)	8938.0 (0.611)	***	30.0%	50.0%	***	6000000	4413616	0	4613372	3573296	1040077
3	11309.0 (0.374)	8791.0 (0.626)	***	65.0%	78.0%	***	2046558	899727	515729	2879243	1970947	908296
4	11117.0 (0.393)	8983.0 (0.607)	***	82.0%	94.0%	***	577312	357689	182950	1672231	927380	744851
5	11248.0 (0.380)	8852.0 (0.620)	***	92.0%	100.0%	***	310886	178379	96362	907746	321936	585809
6	11185.0 (0.387)	8915.0 (0.614)	***	99.0%	100.0%		162153	98232	43467	413541	188254	225286
7	10911.0 (0.414)	9189.0 (0.586)	***	100.0%	100.0%		82016	67280	17827	231440	118614	112825
8	10771.0 (0.428)	9329.0 (0.572)	***	100.0%	100.0%		52300	39541	9031	98327	71416	26911
9	10933.0 (0.412)	9167.0 (0.588)	***	100.0%	100.0%		37576	31934	6866	59250	45423	13827
10	10183.0 (0.487)	9917.0 (0.513)		100.0%	100.0%		25538	25688	654	39191	33489	5702
12	10281.0 (0.477)	9819.0 (0.523)		100.0%	100.0%		15790	15849	700	20885	19022	1863
14	10591.0 (0.446)	9509.0 (0.554)		100.0%	100.0%		11692	10360	1038	12797	11352	1445
16	10837.5 (0.421)	9262.5 (0.579)	***	100.0%	100.0%		7566	7029	838	8627	7551	1076
18	10619.5 (0.443)	9480.5 (0.557)		100.0%	100.0%		5975	5392	478	6376	5827	549
20	10411.5 (0.464)	9688.5 (0.536)		100.0%	100.0%		4411	4208	205	4952	4655	297
30	9261.0 (0.579)	10839.0 (0.421)	***	100.0%	100.0%		1748	1940	-148	1897	2038	-141
40	8688.5 (0.636)	11411.5 (0.364)	***	100.0%	100.0%		1084	1186	-113	1138	1254	-116
50	7831.0 (0.722)	12269.0 (0.278)	***	100.0%	100.0%		766	874	-105	796	898	-102
60	7201.0 (0.785)	12899.0 (0.215)	***	100.0%	100.0%		604	705	-94	624	719	-95
70	6868.0 (0.818)	13232.0 (0.182)	***	100.0%	100.0%		516	603	-83	524	606	-81
80	6464.5 (0.859)	13635.5 (0.141)	***	100.0%	100.0%		454	520	-69	458	527	-69
90	6001.5 (0.905)	14098.5 (0.095)	***	100.0%	100.0%		410	482	-68	412	479	-67
100	5628.0 (0.942)	14472.0 (0.058)	***	100.0%	100.0%		382	451	-68	381	448	-67
120	5594.0 (0.946)	14506.0 (0.054)	***	100.0%	100.0%		353	413	-61	350	411	-61
140	5647.5 (0.940)	14452.5 (0.060)	***	100.0%	100.0%		324	378	-54	324	378	-54
160	5776.0 (0.927)	14324.0 (0.073)	***	100.0%	100.0%		300	348	-49	299	347	-48
180	5946.5 (0.910)	14153.5 (0.090)	***	100.0%	100.0%		278	320	-42	277	319	-42
200	6179.5 (0.887)	13920.5 (0.113)	***	100.0%	100.0%		258	292	-36	257	293	-36
220	6440.5 (0.861)	13659.5 (0.139)	***	100.0%	100.0%		238	267	-31	237	267	-31
240	6730.0 (0.832)	13370.0 (0.168)	***	100.0%	100.0%		216	244	-27	217	243	-26
260	6852.0 (0.820)	13248.0 (0.180)	***	100.0%	100.0%		198	222	-25	198	222	-24
280	7052.0 (0.800)	13048.0 (0.200)	***	100.0%	100.0%		182	202	-22	181	202	-21
300	7278.5 (0.777)	12821.5 (0.223)	***	100.0%	100.0%		165	184	-20	164	183	-19
320	7596.5 (0.745)	12503.5 (0.255)	***	100.0%	100.0%		148	164	-16	148	164	-16
340	7892.0 (0.716)	12208.0 (0.284)	***	100.0%	100.0%		132	146	-13	132	145	-13
360	8203.5 (0.685)	11896.5 (0.315)	***	100.0%	100.0%		117	126	-11	116	127	-11
380	8481.5 (0.657)	11618.5 (0.343)	***	100.0%	100.0%		102	109	-9	101	110	-9
400	8615.5 (0.643)	11484.5 (0.357)	***	100.0%	100.0%		87	93	-8	86	94	-8
420	8974.5 (0.608)	11125.5 (0.392)	***	100.0%	100.0%		73	77	-6	72	78	-6
440	9095.0 (0.596)	11005.0 (0.405)	***	100.0%	100.0%		59	63	-5	58	63	-5
460	9248.0 (0.580)	10852.0 (0.420)	***	100.0%	100.0%		45	49	-4	44	48	-4
480	9576.0 (0.547)	10524.0 (0.453)		100.0%	100.0%		33	35	-2	32	34	-2
500	9866.5 (0.518)	10233.5 (0.482)		100.0%	100.0%		19	21	-1	19	20	-1

表 8：RandomWalkProb=0.300 --- RandomWalkProb=0.400 性能比较统计分析

Quality Bound	RankSum1 (BetterProb)	RankSum2 (BetterProb)	H.T. Sig.	Success Rate1	Success Rate2	H.T. Sig.	Median Time1	Median Time2	Differ Median	Mean Time1	Mean Time2	Mean Diff
0	10241.0 (0.481)	9859.0 (0.519)		4.0%	8.0%		6000000	6000000	0	5801660	5671694	129966
1	10870.0 (0.418)	9230.0 (0.582)	***	18.0%	34.0%	***	6000000	6000000	0	5205951	4471167	734784
2	11394.0 (0.366)	8706.0 (0.634)	***	50.0%	73.0%	***	4413616	1227746	463222	3573296	2328300	1244996
3	11406.0 (0.364)	8694.0 (0.636)	***	78.0%	88.0%	***	899727	421648	231918	1970947	1242882	728065
4	11440.0 (0.361)	8660.0 (0.639)	***	94.0%	97.0%		357689	180675	127216	927380	507239	420141
5	11446.0 (0.360)	8654.0 (0.640)	***	100.0%	100.0%		178379	104272	51165	321936	173725	148211
6	11484.0 (0.357)	8616.0 (0.643)	***	100.0%	100.0%		98232	68294	29642	188254	92590	95665
7	11290.5 (0.376)	8809.5 (0.624)	***	100.0%	100.0%		67280	47862	15300	118614	64431	54183
8	11107.0 (0.394)	8993.0 (0.606)	***	100.0%	100.0%		39541	34250	8926	71416	47533	23883
9	10872.0 (0.418)	9228.0 (0.582)	***	100.0%	100.0%		31934	25986	4996	45423	38134	7288
10	11004.0 (0.405)	9096.0 (0.595)	***	100.0%	100.0%		25688	20844	4564	33489	27218	6271
12	10510.0 (0.454)	9590.0 (0.546)		100.0%	100.0%		15849	13866	1280	19022	18654	368
14	10195.0 (0.485)	9905.0 (0.514)		100.0%	100.0%		10360	9778	258	11352	11681	-329
16	9567.5 (0.548)	10532.5 (0.452)		100.0%	100.0%		7029	7632	-580	7551	9000	-1450
18	9644.0 (0.541)	10456.0 (0.459)		100.0%	100.0%		5392	5468	-348	5827	6405	-579
20	9493.5 (0.556)	10606.5 (0.444)		100.0%	100.0%		4208	4487	-339	4655	5065	-410
30	9213.5 (0.584)	10886.5 (0.416)	***	100.0%	100.0%		1940	2101	-158	2038	2210	-172
40	8693.0 (0.636)	11407.0 (0.364)	***	100.0%	100.0%		1186	1314	-128	1254	1376	-122
50	7909.5 (0.714)	12190.5 (0.286)	***	100.0%	100.0%		874	1013	-126	898	1018	-120
60	7604.5 (0.745)	12495.5 (0.255)	***	100.0%	100.0%		705	810	-94	719	810	-91
70	7147.5 (0.790)	12952.5 (0.210)	***	100.0%	100.0%		603	679	-80	606	688	-83
80	6372.5 (0.868)	13727.5 (0.132)	***	100.0%	100.0%		520	609	-85	527	610	-83
90	5794.5 (0.926)	14305.5 (0.074)	***	100.0%	100.0%		482	562	-87	479	566	-87
100	5528.5 (0.952)	14571.5 (0.048)	***	100.0%	100.0%		451	533	-83	448	533	-85
120	5740.5 (0.931)	14359.5 (0.069)	***	100.0%	100.0%		413	482	-70	411	482	-71
140	5824.5 (0.923)	14275.5 (0.077)	***	100.0%	100.0%		378	437	-61	378	440	-62
160	6024.0 (0.903)	14076.0 (0.097)	***	100.0%	100.0%		348	397	-50	347	398	-51
180	6186.5 (0.886)	13913.5 (0.114)	***	100.0%	100.0%		320	361	-44	319	364	-45
200	6523.0 (0.853)	13577.0 (0.147)	***	100.0%	100.0%		292	329	-37	293	330	-37
220	6652.5 (0.840)	13447.5 (0.160)	***	100.0%	100.0%		267	302	-33	267	301	-33
240	6776.0 (0.827)	13324.0 (0.173)	***	100.0%	100.0%		244	276	-32	243	275	-31
260	7144.5 (0.791)	12955.5 (0.209)	***	100.0%	100.0%		222	247	-26	222	248	-26
280	7498.5 (0.755)	12601.5 (0.245)	***	100.0%	100.0%		202	222	-20	202	223	-21
300	8002.5 (0.705)	12097.5 (0.295)	***	100.0%	100.0%		184	200	-15	183	199	-16
320	8260.5 (0.679)	11839.5 (0.321)	***	100.0%	100.0%		164	176	-13	164	177	-13
340	8438.5 (0.661)	11661.5 (0.339)	***	100.0%	100.0%		146	156	-10	145	155	-10
360	8446.0 (0.660)	11654.0 (0.340)	***	100.0%	100.0%		126	137	-10	127	137	-10
380	8744.0 (0.631)	11356.0 (0.369)	***	100.0%	100.0%		109	118	-8	110	117	-7
400	9028.0 (0.602)	11072.0 (0.398)	***	100.0%	100.0%		93	100	-6	94	99	-6
420	9137.5 (0.591)	10962.5 (0.409)	***	100.0%	100.0%		77	84	-6	78	83	-5
440	9274.5 (0.578)	10825.5 (0.422)	***	100.0%	100.0%		63	68	-4	63	67	-4
460	9460.5 (0.559)	10639.5 (0.441)		100.0%	100.0%		49	53	-3	48	51	-3
480	9650.0 (0.540)	10450.0 (0.460)		100.0%	100.0%		35	35	-2	34	36	-2
500	9661.0 (0.539)	10439.0 (0.461)		100.0%	100.0%		21	22	-2	20	22	-2

表 9：RandomWalkProb=0.400 --- RandomWalkProb=0.500 性能比较统计分析

Quality Bound	RankSum1 (BetterProb)	RankSum2 (BetterProb)	H.T. Sig.	Success Rate1	Success Rate2	H.T. Sig.	Median Time1	Median Time2	Differ Median	Mean Time1	Mean Time2	Mean Diff
0	11244.0 (0.381)	8856.0 (0.619)	***	8.0%	31.0%	***	6000000	6000000	0	5671694	4577781	1093913
1	12384.0 (0.267)	7716.0 (0.733)	***	34.0%	76.0%	***	6000000	1597706	3078248	4471167	2405157	2066010
2	12272.5 (0.278)	7827.5 (0.722)	***	73.0%	95.0%	***	1227746	440866	609389	2328300	937390	1390910
3	11912.0 (0.314)	8188.0 (0.686)	***	88.0%	99.0%	***	421648	189530	193372	1242882	332668	910215
4	11687.0 (0.336)	8413.0 (0.664)	***	97.0%	100.0%	***	180675	117448	68290	507239	155757	351481
5	11725.0 (0.333)	8375.0 (0.667)	***	100.0%	100.0%		104272	60448	34796	173725	85116	88610
6	11471.0 (0.358)	8629.0 (0.642)	***	100.0%	100.0%		68294	45734	18354	92590	59399	33191
7	11496.5 (0.355)	8603.5 (0.645)	***	100.0%	100.0%		47862	30894	12442	64431	39528	24903
8	11342.5 (0.371)	8757.5 (0.629)	***	100.0%	100.0%		34250	24682	7696	47533	29294	18239
9	11613.5 (0.344)	8486.5 (0.656)	***	100.0%	100.0%		25986	18246	7002	38134	21985	16149
10	11498.0 (0.355)	8602.0 (0.645)	***	100.0%	100.0%		20844	13966	4931	27218	18233	8985
12	11518.0 (0.353)	8582.0 (0.647)	***	100.0%	100.0%		13866	10079	3124	18654	11958	6697
14	11394.0 (0.366)	8706.0 (0.634)	***	100.0%	100.0%		9778	7508	1792	11681	8603	3078
16	11391.0 (0.366)	8709.0 (0.634)	***	100.0%	100.0%		7632	5770	1374	9000	6539	2461
18	10882.5 (0.417)	9217.5 (0.583)	***	100.0%	100.0%		5468	5096	563	6405	5347	1059
20	10708.5 (0.434)	9391.5 (0.566)		100.0%	100.0%		4487	4173	342	5065	4475	590
30	9130.0 (0.592)	10970.0 (0.408)	***	100.0%	100.0%		2101	2330	-174	2210	2352	-141
40	8240.0 (0.681)	11860.0 (0.319)	***	100.0%	100.0%		1314	1500	-173	1376	1541	-165
50	7582.5 (0.747)	12517.5 (0.253)	***	100.0%	100.0%		1013	1163	-146	1018	1165	-148
60	6597.5 (0.845)	13502.5 (0.155)	***	100.0%	100.0%		810	960	-155	810	967	-157
70	6045.0 (0.900)	14055.0 (0.100)	***	100.0%	100.0%		679	842	-161	688	849	-161
80	5594.5 (0.946)	14505.5 (0.054)	***	100.0%	100.0%		609	755	-155	610	769	-159
90	5464.0 (0.959)	14636.0 (0.041)	***	100.0%	100.0%		562	711	-149	566	720	-154
100	5423.0 (0.963)	14677.0 (0.037)	***	100.0%	100.0%		533	671	-141	533	678	-145
120	5489.0 (0.956)	14611.0 (0.044)	***	100.0%	100.0%		482	606	-121	482	604	-122
140	5791.0 (0.926)	14309.0 (0.074)	***	100.0%	100.0%		437	534	-95	440	536	-96
160	5911.5 (0.914)	14188.5 (0.086)	***	100.0%	100.0%		397	478	-80	398	480	-81
180	6174.0 (0.888)	13926.0 (0.112)	***	100.0%	100.0%		361	430	-68	364	432	-68
200	6279.5 (0.877)	13820.5 (0.123)	***	100.0%	100.0%		329	382	-56	330	389	-59
220	6390.0 (0.866)	13710.0 (0.134)	***	100.0%	100.0%		302	347	-48	301	351	-50
240	6944.0 (0.811)	13156.0 (0.189)	***	100.0%	100.0%		276	316	-39	275	314	-39
260	7240.0 (0.781)	12860.0 (0.219)	***	100.0%	100.0%		247	281	-34	248	283	-35
280	7257.0 (0.779)	12843.0 (0.221)	***	100.0%	100.0%		222	251	-30	223	253	-30
300	7475.5 (0.757)	12624.5 (0.243)	***	100.0%	100.0%		200	224	-26	199	225	-26
320	7703.0 (0.735)	12397.0 (0.265)	***	100.0%	100.0%		176	199	-22	177	198	-21
340	7839.5 (0.721)	12260.5 (0.279)	***	100.0%	100.0%		156	176	-19	155	174	-18
360	8087.0 (0.696)	12013.0 (0.304)	***	100.0%	100.0%		137	154	-15	137	152	-15
380	8174.0 (0.688)	11926.0 (0.312)	***	100.0%	100.0%		118	131	-14	117	131	-14
400	8561.0 (0.649)	11539.0 (0.351)	***	100.0%	100.0%		100	109	-10	99	110	-11
420	8937.5 (0.611)	11162.5 (0.389)	***	100.0%	100.0%		84	90	-7	83	90	-7
440	9337.5 (0.571)	10762.5 (0.429)	***	100.0%	100.0%		68	72	-4	67	71	-5
460	9454.5 (0.560)	10645.5 (0.440)		100.0%	100.0%		53	56	-4	51	55	-4
480	9591.5 (0.546)	10508.5 (0.454)		100.0%	100.0%		35	40	-3	36	38	-2
500	9777.0 (0.527)	10323.0 (0.473)		100.0%	100.0%		22	24	-2	22	23	-1

表 10：RandomWalkProb=0.500 --- RandomWalkProb=0.600 性能比较统计分析

Quality Bound	RankSum1 (BetterProb)	RankSum2 (BetterProb)	H.T. Sig.	Success Rate1	Success Rate2	H.T. Sig.	Median Time1	Median Time2	Differ Median	Mean Time1	Mean Time2	Mean Diff
0	10662.5 (0.439)	9437.5 (0.561)		31.0%	43.0%	***	6000000	6000000	0	4577781	4019459	558322
1	12709.0 (0.234)	7391.0 (0.766)	***	76.0%	97.0%	***	1597706	567602	846851	2405157	907256	1497901
2	12704.0 (0.235)	7396.0 (0.765)	***	95.0%	100.0%	***	440866	226292	244156	937390	272285	665105
3	11903.0 (0.315)	8197.0 (0.685)	***	99.0%	100.0%		189530	118676	67021	332668	149425	183243
4	11436.0 (0.361)	8664.0 (0.639)	***	100.0%	100.0%		117448	77179	28594	155757	95030	60727
5	10993.0 (0.406)	9107.0 (0.594)	***	100.0%	100.0%		60448	49188	10997	85116	64684	20431
6	10656.0 (0.439)	9444.0 (0.561)		100.0%	100.0%		45734	38488	5374	59399	46730	12669
7	10268.5 (0.478)	9831.5 (0.522)		100.0%	100.0%		30894	32190	1364	39528	35351	4177
8	9976.0 (0.507)	10124.0 (0.493)		100.0%	100.0%		24682	24977	-380	29294	29233	61
9	9694.0 (0.536)	10406.0 (0.464)		100.0%	100.0%		18246	21876	-1206	21985	22626	-641
10	9835.0 (0.521)	10265.0 (0.478)		100.0%	100.0%		13966	16578	-541	18233	18191	42
12	9148.0 (0.590)	10952.0 (0.410)	***	100.0%	100.0%		10079	11539	-1534	11958	13438	-1480
14	8855.5 (0.619)	11244.5 (0.381)	***	100.0%	100.0%		7508	8826	-1282	8603	9936	-1333
16	8370.0 (0.668)	11730.0 (0.332)	***	100.0%	100.0%		5770	7680	-1342	6539	8049	-1510
18	8351.0 (0.670)	11749.0 (0.330)	***	100.0%	100.0%		5096	6046	-1124	5347	6702	-1355
20	7929.0 (0.712)	12171.0 (0.288)	***	100.0%	100.0%		4173	5411	-1118	4475	5830	-1356
30	7184.5 (0.787)	12915.5 (0.213)	***	100.0%	100.0%		2330	3050	-728	2352	3084	-732
40	6103.5 (0.895)	13996.5 (0.105)	***	100.0%	100.0%		1500	2161	-616	1541	2185	-644
50	5457.5 (0.959)	14642.5 (0.041)	***	100.0%	100.0%		1163	1656	-501	1165	1675	-509
60	5341.0 (0.971)	14759.0 (0.029)	***	100.0%	100.0%		960	1439	-473	967	1438	-471
70	5256.0 (0.979)	14844.0 (0.021)	***	100.0%	100.0%		842	1279	-406	849	1260	-411
80	5217.0 (0.983)	14883.0 (0.017)	***	100.0%	100.0%		755	1093	-339	769	1116	-347
90	5261.0 (0.979)	14839.0 (0.021)	***	100.0%	100.0%		711	1012	-285	720	1007	-287
100	5372.0 (0.968)	14728.0 (0.032)	***	100.0%	100.0%		671	908	-235	678	921	-243
120	5493.0 (0.956)	14607.0 (0.044)	***	100.0%	100.0%		606	775	-178	604	789	-186
140	5776.0 (0.927)	14324.0 (0.073)	***	100.0%	100.0%		534	682	-146	536	684	-147
160	6034.0 (0.902)	14066.0 (0.098)	***	100.0%	100.0%		478	586	-111	480	593	-113
180	6396.5 (0.865)	13703.5 (0.135)	***	100.0%	100.0%		430	520	-88	432	520	-88
200	6528.5 (0.852)	13571.5 (0.148)	***	100.0%	100.0%		382	456	-71	389	459	-70
220	6706.5 (0.834)	13393.5 (0.166)	***	100.0%	100.0%		347	408	-58	351	409	-58
240	7010.5 (0.804)	13089.5 (0.196)	***	100.0%	100.0%		316	359	-50	314	365	-51
260	7418.5 (0.763)	12681.5 (0.237)	***	100.0%	100.0%		281	323	-39	283	322	-39
280	7844.0 (0.721)	12256.0 (0.279)	***	100.0%	100.0%		251	284	-29	253	282	-29
300	8115.0 (0.694)	11985.0 (0.306)	***	100.0%	100.0%		224	248	-22	225	248	-23
320	8366.5 (0.668)	11733.5 (0.332)	***	100.0%	100.0%		199	216	-17	198	216	-18
340	8463.0 (0.659)	11637.0 (0.341)	***	100.0%	100.0%		176	188	-14	174	189	-16
360	8693.5 (0.636)	11406.5 (0.364)	***	100.0%	100.0%		154	166	-12	152	164	-12
380	8899.0 (0.615)	11201.0 (0.385)	***	100.0%	100.0%		131	143	-9	131	140	-9
400	8944.5 (0.611)	11155.5 (0.389)	***	100.0%	100.0%		109	120	-9	110	118	-8
420	9215.0 (0.584)	10885.0 (0.416)	***	100.0%	100.0%		90	97	-6	90	96	-6
440	9293.0 (0.576)	10807.0 (0.424)	***	100.0%	100.0%		72	76	-5	71	76	-5
460	9684.0 (0.537)	10416.0 (0.463)		100.0%	100.0%		56	57	-2	55	57	-2
480	9768.0 (0.528)	10332.0 (0.472)		100.0%	100.0%		40	40	-2	38	40	-2
500	9877.0 (0.517)	10223.0 (0.483)		100.0%	100.0%		24	24	-1	23	25	-1

表 11：RandomWalkProb=0.600 --- RandomWalkProb=0.640 性能比较统计分析

Quality Bound	RankSum1 (BetterProb)	RankSum2 (BetterProb)	H.T. Sig.	Success Rate1	Success Rate2	H.T. Sig.	Median Time1	Median Time2	Differ Median	Mean Time1	Mean Time2	Mean Diff
0	9272.5 (0.578)	10827.5 (0.422)	***	43.0%	27.0%	***	6000000	6000000	0	4019459	4741452	-721993
1	8532.5 (0.652)	11567.5 (0.348)	***	97.0%	89.0%	***	567602	1063022	-337146	907256	1591820	-684565
2	8262.0 (0.679)	11838.0 (0.321)	***	100.0%	100.0%		226292	388862	-141240	272285	469915	-197630
3	8772.0 (0.628)	11328.0 (0.372)	***	100.0%	100.0%		118676	180788	-46250	149425	217131	-67705
4	9120.0 (0.593)	10980.0 (0.407)	***	100.0%	100.0%		77179	96484	-17809	95030	127396	-32366
5	9000.0 (0.605)	11100.0 (0.395)	***	100.0%	100.0%		49188	68668	-14402	64684	79057	-14373
6	8958.0 (0.609)	11142.0 (0.391)	***	100.0%	100.0%		38488	53554	-10436	46730	60190	-13460
7	9002.0 (0.605)	11098.0 (0.395)	***	100.0%	100.0%		32190	38738	-6682	35351	45172	-9821
8	8910.0 (0.614)	11190.0 (0.386)	***	100.0%	100.0%		24977	31392	-5666	29233	35443	-6210
9	8793.0 (0.626)	11307.0 (0.374)	***	100.0%	100.0%		21876	25676	-5046	22626	28560	-5934
10	8346.0 (0.670)	11754.0 (0.330)	***	100.0%	100.0%		16578	21008	-4996	18191	24703	-6512
12	8302.0 (0.675)	11798.0 (0.325)	***	100.0%	100.0%		11539	16171	-3506	13438	17287	-3849
14	8177.0 (0.687)	11923.0 (0.313)	***	100.0%	100.0%		8826	11955	-2652	9936	13135	-3199
16	7903.5 (0.715)	12196.5 (0.285)	***	100.0%	100.0%		7680	10128	-2598	8049	10694	-2645
18	7962.0 (0.709)	12138.0 (0.291)	***	100.0%	100.0%		6046	8369	-1906	6702	8757	-2056
20	8004.0 (0.705)	12096.0 (0.295)	***	100.0%	100.0%		5411	6817	-1444	5830	7384	-1554
30	7503.5 (0.755)	12596.5 (0.245)	***	100.0%	100.0%		3050	3805	-824	3084	4015	-931
40	6922.5 (0.813)	13177.5 (0.187)	***	100.0%	100.0%		2161	2761	-606	2185	2811	-626
50	6355.5 (0.869)	13744.5 (0.131)	***	100.0%	100.0%		1656	2138	-521	1675	2243	-568
60	6765.5 (0.828)	13334.5 (0.172)	***	100.0%	100.0%		1439	1822	-398	1438	1853	-415
70	6721.5 (0.833)	13378.5 (0.167)	***	100.0%	100.0%		1279	1573	-321	1260	1608	-348
80	6586.5 (0.846)	13513.5 (0.154)	***	100.0%	100.0%		1093	1372	-272	1116	1402	-286
90	6703.5 (0.835)	13396.5 (0.165)	***	100.0%	100.0%		1012	1227	-220	1007	1244	-237
100	6717.0 (0.833)	13383.0 (0.167)	***	100.0%	100.0%		908	1100	-197	921	1129	-209
120	7129.5 (0.792)	12970.5 (0.208)	***	100.0%	100.0%		775	926	-135	789	931	-142
140	7668.0 (0.738)	12432.0 (0.262)	***	100.0%	100.0%		682	786	-91	684	778	-95
160	7917.5 (0.713)	12182.5 (0.287)	***	100.0%	100.0%		586	651	-65	593	664	-72
180	8011.0 (0.704)	12089.0 (0.296)	***	100.0%	100.0%		520	562	-52	520	579	-58
200	8052.0 (0.700)	12048.0 (0.300)	***	100.0%	100.0%		456	494	-38	459	500	-41
220	8415.0 (0.663)	11685.0 (0.337)	***	100.0%	100.0%		408	436	-29	409	439	-30
240	8710.0 (0.634)	11390.0 (0.366)	***	100.0%	100.0%		359	384	-22	365	388	-23
260	8690.5 (0.636)	11409.5 (0.364)	***	100.0%	100.0%		323	338	-19	322	342	-20
280	8851.0 (0.620)	11249.0 (0.380)	***	100.0%	100.0%		284	296	-17	282	300	-18
300	9102.5 (0.595)	10997.5 (0.405)	***	100.0%	100.0%		248	257	-12	248	260	-12
320	9058.5 (0.599)	11041.5 (0.401)	***	100.0%	100.0%		216	224	-11	216	228	-12
340	9014.5 (0.604)	11085.5 (0.396)	***	100.0%	100.0%		188	197	-10	189	199	-10
360	9062.0 (0.599)	11038.0 (0.401)	***	100.0%	100.0%		166	173	-9	164	173	-9
380	9285.0 (0.577)	10815.0 (0.423)	***	100.0%	100.0%		143	148	-7	140	147	-7
400	9334.5 (0.572)	10765.5 (0.428)	***	100.0%	100.0%		120	127	-6	118	124	-5
420	9431.5 (0.562)	10668.5 (0.438)		100.0%	100.0%		97	102	-5	96	100	-4
440	9515.5 (0.553)	10584.5 (0.447)		100.0%	100.0%		76	82	-4	76	79	-3
460	9502.5 (0.555)	10597.5 (0.445)		100.0%	100.0%		57	64	-4	57	60	-3
480	9619.0 (0.543)	10481.0 (0.457)		100.0%	100.0%		40	45	-3	40	43	-2
500	9869.0 (0.518)	10231.0 (0.482)		100.0%	100.0%		24	27	-1	25	25	-1

表 12：RandomWalkProb=0.640 --- RandomWalkProb=0.700 性能比较统计分析

Quality Bound	RankSum1 (BetterProb)	RankSum2 (BetterProb)	H.T. Sig.	Success Rate1	Success Rate2	H.T. Sig.	Median Time1	Median Time2	Differ Median	Mean Time1	Mean Time2	Mean Diff
0	8700.0 (0.635)	11400.0 (0.365)	***	27.0%	0.0%	***	6000000	6000000	0	4741452	6000000	-1258548
1	5600.0 (0.945)	14500.0 (0.055)	***	89.0%	0.0%	***	1063022	6000000	-4936978	1591820	6000000	-4408180
2	5050.0 (1.000)	15050.0 (0.000)	***	100.0%	0.0%	***	388862	6000000	-5611138	469915	6000000	-5530085
3	5050.0 (1.000)	15050.0 (0.000)	***	100.0%	2.0%	***	180788	6000000	-5810080	217131	5903239	-5686108
4	5050.0 (1.000)	15050.0 (0.000)	***	100.0%	7.0%	***	96484	6000000	-5898631	127396	5697657	-5570261
5	5179.0 (0.987)	14921.0 (0.013)	***	100.0%	19.0%	***	68668	6000000	-5917340	79057	5127674	-5048616
6	5175.0 (0.988)	14925.0 (0.013)	***	100.0%	29.0%	***	53554	6000000	-5931152	60190	4607828	-4547638
7	5146.0 (0.990)	14954.0 (0.010)	***	100.0%	54.0%	***	38738	2467986	-2427210	45172	3357618	-3312446
8	5229.0 (0.982)	14871.0 (0.018)	***	100.0%	86.0%	***	31392	922817	-872750	35443	1705934	-1670491
9	5192.0 (0.986)	14908.0 (0.014)	***	100.0%	99.0%		25676	544916	-502940	28560	785149	-756588
10	5255.0 (0.980)	14845.0 (0.021)	***	100.0%	100.0%		21008	288940	-266328	24703	388292	-363589
12	5224.0 (0.983)	14876.0 (0.017)	***	100.0%	100.0%		16171	165913	-149852	17287	206095	-188808
14	5240.0 (0.981)	14860.0 (0.019)	***	100.0%	100.0%		11955	70998	-58427	13135	101112	-87977
16	5240.0 (0.981)	14860.0 (0.019)	***	100.0%	100.0%		10128	41830	-31440	10694	52476	-41782
18	5367.0 (0.968)	14733.0 (0.032)	***	100.0%	100.0%		8369	28774	-21192	8757	37202	-28444
20	5406.0 (0.964)	14694.0 (0.036)	***	100.0%	100.0%		6817	22561	-15761	7384	29017	-21632
30	5500.0 (0.955)	14600.0 (0.045)	***	100.0%	100.0%		3805	8474	-4706	4015	9519	-5504
40	5529.0 (0.952)	14571.0 (0.048)	***	100.0%	100.0%		2761	5276	-2512	2811	5588	-2777
50	5739.0 (0.931)	14361.0 (0.069)	***	100.0%	100.0%		2138	3409	-1264	2243	3812	-1569
60	5794.0 (0.926)	14306.0 (0.074)	***	100.0%	100.0%		1822	2749	-939	1853	2912	-1059
70	6100.0 (0.895)	14000.0 (0.105)	***	100.0%	100.0%		1573	2312	-696	1608	2363	-755
80	6067.0 (0.898)	14033.0 (0.102)	***	100.0%	100.0%		1372	1930	-560	1402	1994	-593
90	6147.0 (0.890)	13953.0 (0.110)	***	100.0%	100.0%		1227	1682	-458	1244	1727	-483
100	6210.0 (0.884)	13890.0 (0.116)	***	100.0%	100.0%		1100	1474	-350	1129	1509	-379
120	6264.5 (0.879)	13835.5 (0.121)	***	100.0%	100.0%		926	1144	-236	931	1184	-253
140	6491.5 (0.856)	13608.5 (0.144)	***	100.0%	100.0%		786	945	-179	778	964	-186
160	6832.0 (0.822)	13268.0 (0.178)	***	100.0%	100.0%		651	800	-138	664	799	-134
180	6934.5 (0.812)	13165.5 (0.188)	***	100.0%	100.0%		562	666	-101	579	681	-102
200	6924.0 (0.813)	13176.0 (0.187)	***	100.0%	100.0%		494	576	-76	500	576	-76
220	7358.0 (0.769)	12742.0 (0.231)	***	100.0%	100.0%		436	488	-56	439	497	-58
240	7631.5 (0.742)	12468.5 (0.258)	***	100.0%	100.0%		384	428	-46	388	433	-45
260	8007.0 (0.704)	12093.0 (0.296)	***	100.0%	100.0%		338	369	-32	342	375	-33
280	8350.5 (0.670)	11749.5 (0.330)	***	100.0%	100.0%		296	322	-26	300	327	-27
300	8392.0 (0.666)	11708.0 (0.334)	***	100.0%	100.0%		257	279	-22	260	283	-22
320	8590.0 (0.646)	11510.0 (0.354)	***	100.0%	100.0%		224	241	-16	228	245	-16
340	8846.5 (0.620)	11253.5 (0.380)	***	100.0%	100.0%		197	209	-13	199	213	-14
360	9088.0 (0.596)	11012.0 (0.404)	***	100.0%	100.0%		173	181	-9	173	183	-11
380	9200.0 (0.585)	10900.0 (0.415)	***	100.0%	100.0%		148	155	-8	147	156	-9
400	9569.0 (0.548)	10531.0 (0.452)		100.0%	100.0%		127	129	-4	124	129	-6
420	9431.0 (0.562)	10669.0 (0.438)		100.0%	100.0%		102	108	-5	100	107	-6
440	9730.5 (0.532)	10369.5 (0.468)		100.0%	100.0%		82	81	-3	79	84	-4
460	9956.5 (0.509)	10143.5 (0.491)		100.0%	100.0%		64	62	-0	60	62	-2
480	9972.0 (0.508)	10128.0 (0.492)		100.0%	100.0%		45	43	-0	43	44	-1
500	10072.5 (0.498)	10027.5 (0.502)		100.0%	100.0%		27	23	0	25	26	-0

表 13：RandomWalkProb=0.000，随机邻域搜索 --- 环形邻域搜索

Quality Bound	RankSum1 (BetterProb)	RankSum2 (BetterProb)	H.T. Sig.	Success Rate1	Success Rate2	H.T. Sig.	Median Time1	Median Time2	Differ Median	Mean Time1	Mean Time2	Mean Diff
0	10050.0 (0.500)	10050.0 (0.500)		0.0%	0.0%		6000000	6000000	0	6000000	6000000	0
1	10100.0 (0.495)	10000.0 (0.505)		0.0%	1.0%		6000000	6000000	0	6000000	5940502	59498
2	10150.5 (0.490)	9949.5 (0.510)		1.0%	3.0%		6000000	6000000	0	5943333	5836747	106586
3	10168.0 (0.488)	9932.0 (0.512)		10.0%	12.0%		6000000	6000000	0	5502414	5366355	136059
4	10316.0 (0.473)	9784.0 (0.527)		20.0%	24.0%		6000000	6000000	0	4950575	4688962	261614
5	11010.0 (0.404)	9090.0 (0.596)	***	26.0%	44.0%	***	6000000	6000000	0	4542375	3543891	998485
6	11149.5 (0.390)	8950.5 (0.610)	***	45.0%	61.0%	***	6000000	382292	6029	3467461	2529249	938212
7	11644.0 (0.341)	8456.0 (0.659)	***	62.0%	79.0%	***	460021	136659	82772	2479942	1470457	1009485
8	12303.0 (0.275)	7797.0 (0.725)	***	81.0%	88.0%		118886	29833	64504	1341408	849923	491485
9	12678.0 (0.237)	7422.0 (0.763)	***	87.0%	93.0%		60938	17278	36464	928852	563083	365768
10	13553.0 (0.150)	6547.0 (0.850)	***	90.0%	99.0%	***	43734	11174	29204	717257	81759	635498
12	13999.0 (0.105)	6101.0 (0.895)	***	98.0%	100.0%		23420	6206	16394	178806	10434	168371
14	14338.0 (0.071)	5762.0 (0.929)	***	99.0%	100.0%		14522	4143	10160	89320	5059	84261
16	14177.0 (0.087)	5923.0 (0.913)	***	100.0%	100.0%		9148	3338	5710	13085	3711	9374
18	14201.5 (0.085)	5898.5 (0.915)	***	100.0%	100.0%		6796	2684	3878	7989	2862	5127
20	14075.5 (0.097)	6024.5 (0.903)	***	100.0%	100.0%		5076	2288	2549	5682	2402	3280
30	12289.0 (0.276)	7811.0 (0.724)	***	100.0%	100.0%		1742	1512	290	1849	1489	360
40	7689.5 (0.736)	12410.5 (0.264)	***	100.0%	100.0%		998	1189	-180	1030	1184	-154
50	5451.5 (0.960)	14648.5 (0.040)	***	100.0%	100.0%		672	1010	-328	706	1018	-312
60	5051.0 (1.000)	15049.0 (0.000)	***	100.0%	100.0%		536	910	-371	540	907	-367
70	5050.0 (1.000)	15050.0 (0.000)	***	100.0%	100.0%		438	841	-390	445	834	-388
80	5050.0 (1.000)	15050.0 (0.000)	***	100.0%	100.0%		380	770	-392	379	771	-392
90	5050.0 (1.000)	15050.0 (0.000)	***	100.0%	100.0%		337	719	-382	337	717	-380
100	5050.0 (1.000)	15050.0 (0.000)	***	100.0%	100.0%		305	678	-368	307	674	-367
120	5050.0 (1.000)	15050.0 (0.000)	***	100.0%	100.0%		279	604	-324	278	599	-322
140	5050.0 (1.000)	15050.0 (0.000)	***	100.0%	100.0%		262	544	-281	260	539	-278
160	5050.0 (1.000)	15050.0 (0.000)	***	100.0%	100.0%		244	490	-246	244	487	-244
180	5050.0 (1.000)	15050.0 (0.000)	***	100.0%	100.0%		228	446	-217	228	443	-216
200	5050.0 (1.000)	15050.0 (0.000)	***	100.0%	100.0%		214	407	-195	212	406	-194
220	5050.0 (1.000)	15050.0 (0.000)	***	100.0%	100.0%		200	374	-178	198	375	-177
240	5050.0 (1.000)	15050.0 (0.000)	***	100.0%	100.0%		184	345	-161	183	343	-160
260	5050.0 (1.000)	15050.0 (0.000)	***	100.0%	100.0%		170	314	-146	169	314	-145
280	5050.0 (1.000)	15050.0 (0.000)	***	100.0%	100.0%		156	287	-132	155	285	-130
300	5050.0 (1.000)	15050.0 (0.000)	***	100.0%	100.0%		142	260	-118	141	258	-117
320	5050.0 (1.000)	15050.0 (0.000)	***	100.0%	100.0%		129	236	-105	127	231	-103
340	5050.0 (1.000)	15050.0 (0.000)	***	100.0%	100.0%		116	204	-91	115	204	-90
360	5058.0 (0.999)	15042.0 (0.001)	***	100.0%	100.0%		104	181	-78	102	179	-77
380	5086.5 (0.996)	15013.5 (0.004)	***	100.0%	100.0%		90	158	-68	89	156	-67
400	5206.0 (0.984)	14894.0 (0.016)	***	100.0%	100.0%		78	134	-57	77	133	-56
420	5421.5 (0.963)	14678.5 (0.037)	***	100.0%	100.0%		66	114	-49	64	112	-48
440	5683.0 (0.937)	14417.0 (0.063)	***	100.0%	100.0%		54	94	-40	52	91	-39
460	6320.5 (0.873)	13779.5 (0.127)	***	100.0%	100.0%		42	71	-30	40	70	-29
480	7189.0 (0.786)	12911.0 (0.214)	***	100.0%	100.0%		30	52	-20	28	48	-20
500	8130.5 (0.692)	11969.5 (0.308)	***	100.0%	100.0%		18	29	-12	17	28	-11

表 14：环形邻域搜索，RandomWalkProb=0.000 --- RandomWalkProb=0.037

Quality Bound	RankSum1 (BetterProb)	RankSum2 (BetterProb)	H.T. Sig.	Success Rate1	Success Rate2	H.T. Sig.	Median Time1	Median Time2	Differ Median	Mean Time1	Mean Time2	Mean Diff
0	10350.0 (0.470)	9750.0 (0.530)		0.0%	6.0%	***	6000000	6000000	0	6000000	5741886	258114
1	12821.5 (0.223)	7278.5 (0.777)	***	1.0%	57.0%	***	6000000	2068589	3913522	5940502	3305491	2635011
2	14655.0 (0.040)	5445.0 (0.961)	***	3.0%	96.0%	***	6000000	467247	5473076	5836747	911801	4924946
3	14616.0 (0.043)	5484.0 (0.957)	***	12.0%	100.0%	***	6000000	139972	5843036	5366355	210585	5155770
4	14393.0 (0.066)	5707.0 (0.934)	***	24.0%	100.0%	***	6000000	81006	5904388	4688962	110847	4578114
5	14003.0 (0.105)	6097.0 (0.895)	***	44.0%	100.0%	***	6000000	46850	5887051	3543891	58735	3485156
6	13192.0 (0.186)	6908.0 (0.814)	***	61.0%	100.0%	***	382292	35328	330446	2529249	41078	2488172
7	12530.0 (0.252)	7570.0 (0.748)	***	79.0%	100.0%	***	136659	24574	100776	1470457	27546	1442911
8	11498.5 (0.355)	8601.5 (0.645)	***	88.0%	100.0%	***	29833	17656	11082	849923	22142	827781
9	10698.0 (0.435)	9402.0 (0.565)		93.0%	100.0%	***	17278	14160	2651	563083	17009	546074
10	10004.0 (0.505)	10096.0 (0.495)		99.0%	100.0%		11174	11730	-118	81759	13637	68122
12	8928.0 (0.612)	11172.0 (0.388)	***	100.0%	100.0%		6206	7481	-1282	10434	8594	1840
14	7883.5 (0.717)	12216.5 (0.283)	***	100.0%	100.0%		4143	6060	-1741	5059	6798	-1739
16	7412.0 (0.764)	12688.0 (0.236)	***	100.0%	100.0%		3338	4936	-1514	3711	5243	-1532
18	6825.5 (0.822)	13274.5 (0.178)	***	100.0%	100.0%		2684	4122	-1293	2862	4346	-1484
20	6583.0 (0.847)	13517.0 (0.153)	***	100.0%	100.0%		2288	3228	-981	2402	3536	-1134
30	5896.5 (0.915)	14203.5 (0.085)	***	100.0%	100.0%		1512	1983	-494	1489	2024	-536
40	5644.0 (0.941)	14456.0 (0.059)	***	100.0%	100.0%		1189	1474	-290	1184	1492	-308
50	5903.0 (0.915)	14197.0 (0.085)	***	100.0%	100.0%		1010	1223	-200	1018	1222	-204
60	6094.0 (0.896)	14006.0 (0.104)	***	100.0%	100.0%		910	1051	-152	907	1055	-148
70	6190.5 (0.886)	13909.5 (0.114)	***	100.0%	100.0%		841	948	-115	834	951	-117
80	6482.0 (0.857)	13618.0 (0.143)	***	100.0%	100.0%		770	868	-93	771	863	-92
90	6481.5 (0.857)	13618.5 (0.143)	***	100.0%	100.0%		719	798	-80	717	797	-80
100	6843.0 (0.821)	13257.0 (0.179)	***	100.0%	100.0%		678	738	-65	674	741	-67
120	6948.0 (0.810)	13152.0 (0.190)	***	100.0%	100.0%		604	655	-54	599	654	-55
140	7169.5 (0.788)	12930.5 (0.212)	***	100.0%	100.0%		544	582	-43	539	582	-43
160	7558.5 (0.749)	12541.5 (0.251)	***	100.0%	100.0%		490	526	-34	487	521	-34
180	7856.5 (0.719)	12243.5 (0.281)	***	100.0%	100.0%		446	474	-26	443	470	-26
200	7979.5 (0.707)	12120.5 (0.293)	***	100.0%	100.0%		407	429	-23	406	429	-23
220	8312.0 (0.674)	11788.0 (0.326)	***	100.0%	100.0%		374	393	-18	375	393	-18
240	8482.5 (0.657)	11617.5 (0.343)	***	100.0%	100.0%		345	359	-16	343	359	-16
260	8612.5 (0.644)	11487.5 (0.356)	***	100.0%	100.0%		314	328	-14	314	327	-14
280	8773.0 (0.628)	11327.0 (0.372)	***	100.0%	100.0%		287	297	-12	285	297	-12
300	8928.5 (0.612)	11171.5 (0.388)	***	100.0%	100.0%		260	271	-10	258	268	-10
320	8976.5 (0.607)	11123.5 (0.393)	***	100.0%	100.0%		236	243	-10	231	241	-10
340	9243.5 (0.581)	10856.5 (0.419)	***	100.0%	100.0%		204	215	-7	204	212	-8
360	9199.0 (0.585)	10901.0 (0.415)	***	100.0%	100.0%		181	188	-7	179	186	-7
380	9402.5 (0.565)	10697.5 (0.435)		100.0%	100.0%		158	164	-5	156	161	-5
400	9478.0 (0.557)	10622.0 (0.443)		100.0%	100.0%		134	139	-4	133	138	-5
420	9534.5 (0.552)	10565.5 (0.448)		100.0%	100.0%		114	118	-4	112	116	-4
440	9684.5 (0.537)	10415.5 (0.463)		100.0%	100.0%		94	95	-2	91	94	-3
460	9705.5 (0.534)	10394.5 (0.466)		100.0%	100.0%		71	74	-2	70	72	-3
480	9821.5 (0.523)	10278.5 (0.477)		100.0%	100.0%		52	52	-2	48	50	-2
500	9964.0 (0.509)	10136.0 (0.491)		100.0%	100.0%		29	29	0	28	29	-1

表 15：　　随机邻域搜索，RandomWalkProb=0.600
　　---　环行邻域搜索，RandomWalkProb=0.037

Quality Bound	RankSum1 (BetterProb)	RankSum2 (BetterProb)	H.T. Sig.	Success Rate1	Success Rate2	H.T. Sig.	Median Time1	Median Time2	Differ Median	Mean Time1	Mean Time2	Mean Diff
0	8170.0 (0.688)	11930.0 (0.312)	***	43.0%	6.0%	***	6000000	6000000	0	4019459	5741886	-1722427
1	6887.5 (0.816)	13212.5 (0.184)	***	97.0%	57.0%	***	567602	2068589	-1503082	907256	3305491	-2398236
2	7742.0 (0.731)	12358.0 (0.269)	***	100.0%	96.0%	***	226292	467247	-262176	272285	911801	-639516
3	9335.0 (0.572)	10765.0 (0.428)	***	100.0%	100.0%		118676	139972	-22282	149425	210585	-61160
4	10162.0 (0.489)	9938.0 (0.511)		100.0%	100.0%		77179	81006	1996	95030	110847	-15817
5	10561.0 (0.449)	9539.0 (0.551)		100.0%	100.0%		49188	46850	5387	64684	58735	5950
6	10758.0 (0.429)	9342.0 (0.571)	***	100.0%	100.0%		38488	35328	5624	46730	41078	5652
7	11352.5 (0.370)	8747.5 (0.630)	***	100.0%	100.0%		32190	24574	6706	35351	27546	7805
8	11398.5 (0.365)	8701.5 (0.635)	***	100.0%	100.0%		24977	17656	5742	29233	22142	7091
9	11679.0 (0.337)	8421.0 (0.663)	***	100.0%	100.0%		21876	14160	5499	22626	17009	5617
10	11815.0 (0.324)	8285.0 (0.676)	***	100.0%	100.0%		16578	11730	4131	18191	13637	4554
12	12793.5 (0.226)	7306.5 (0.774)	***	100.0%	100.0%		11539	7481	4161	13438	8594	4844
14	12736.5 (0.231)	7363.5 (0.769)	***	100.0%	100.0%		8826	6060	2809	9936	6798	3138
16	12969.0 (0.208)	7131.0 (0.792)	***	100.0%	100.0%		7680	4936	2428	8049	5243	2806
18	13128.0 (0.192)	6972.0 (0.808)	***	100.0%	100.0%		6046	4122	2046	6702	4346	2355
20	13722.5 (0.133)	6377.5 (0.867)	***	100.0%	100.0%		5411	3228	1976	5830	3536	2295
30	14075.0 (0.098)	6025.0 (0.902)	***	100.0%	100.0%		3050	1983	1032	3084	2024	1059
40	14384.0 (0.067)	5716.0 (0.933)	***	100.0%	100.0%		2161	1474	652	2185	1492	693
50	14464.0 (0.059)	5636.0 (0.941)	***	100.0%	100.0%		1656	1223	442	1675	1222	453
60	14466.0 (0.058)	5634.0 (0.942)	***	100.0%	100.0%		1439	1051	384	1438	1055	383
70	14453.5 (0.060)	5646.5 (0.940)	***	100.0%	100.0%		1279	948	306	1260	951	309
80	14492.5 (0.056)	5607.5 (0.944)	***	100.0%	100.0%		1093	868	242	1116	863	253
90	14515.0 (0.053)	5585.0 (0.947)	***	100.0%	100.0%		1012	798	210	1007	797	210
100	14325.0 (0.072)	5775.0 (0.927)	***	100.0%	100.0%		908	738	169	921	741	180
120	13932.0 (0.112)	6168.0 (0.888)	***	100.0%	100.0%		775	655	126	789	654	135
140	13646.0 (0.140)	6454.0 (0.860)	***	100.0%	100.0%		682	582	99	684	582	102
160	13135.5 (0.191)	6964.5 (0.809)	***	100.0%	100.0%		586	526	67	593	521	71
180	12567.0 (0.248)	7533.0 (0.752)	***	100.0%	100.0%		520	474	49	520	470	51
200	11907.5 (0.314)	8192.5 (0.686)	***	100.0%	100.0%		456	429	29	459	429	29
220	11043.0 (0.401)	9057.0 (0.599)	***	100.0%	100.0%		408	393	14	409	393	15
240	10326.5 (0.472)	9773.5 (0.528)		100.0%	100.0%		359	359	4	365	359	6
260	9498.5 (0.555)	10601.5 (0.445)		100.0%	100.0%		323	328	-7	322	327	-6
280	8694.5 (0.636)	11405.5 (0.364)	***	100.0%	100.0%		284	297	-16	282	297	-15
300	8124.5 (0.693)	11975.5 (0.307)	***	100.0%	100.0%		248	271	-21	248	268	-20
320	7736.5 (0.731)	12363.5 (0.269)	***	100.0%	100.0%		216	243	-26	216	241	-24
340	7744.0 (0.731)	12356.0 (0.269)	***	100.0%	100.0%		188	215	-24	189	212	-23
360	7741.0 (0.731)	12359.0 (0.269)	***	100.0%	100.0%		166	188	-22	164	186	-22
380	7696.5 (0.735)	12403.5 (0.265)	***	100.0%	100.0%		143	164	-21	140	161	-21
400	7749.5 (0.730)	12350.5 (0.270)	***	100.0%	100.0%		120	139	-19	118	138	-19
420	7649.0 (0.740)	12451.0 (0.260)	***	100.0%	100.0%		97	118	-20	96	116	-19
440	7837.0 (0.721)	12263.0 (0.279)	***	100.0%	100.0%		76	95	-18	76	94	-17
460	8127.5 (0.692)	11972.5 (0.308)	***	100.0%	100.0%		57	74	-15	57	72	-15
480	8696.0 (0.635)	11404.0 (0.365)	***	100.0%	100.0%		40	52	-10	40	50	-9
500	9323.5 (0.573)	10776.5 (0.427)	***	100.0%	100.0%		24	29	-5	25	29	-4

第三章　局部搜索多初始点选择的划分策略的性能分析

§1. 简介

局部搜索(Local Search)是一种求解组合优化问题的通用和实用的启发式方法。早在七十年代，Lin 和 Kernighan 用局部搜索技术近似求解图划分问题(Graph Partitioning Problem)[Kernighan & Lin, 1970]和旅行商问题(Traveling Salesman Problem) [Lin & Kernighan, 1973]取得了成功，引起了研究者对局部搜索技术的广泛关注[Papadimitriou & Steiglitz, 1982]。九十年代，Selman、顾钧(J. Gu)等多位学者用局部搜索技术求解可满足性问题(Satisfiability Problem)取得了引人注目的实验成果[Selman *et al.*, 1992, 1994; Gu, 1992, 1993, 1994]。近些年来比较流行的模拟退火(Simulated Annealing) 算法 [Kirpatrick *et al.*, 1983] 、遗传算法 (Genetic Algorithm)[Goldberg, 1989]等启发式方法均可看成是局部搜索方法的进一步发展。此外，局部搜索技术与连续问题最优化方法也有深刻的联系。局部搜索在方法上的一般性和应用上的有效性促进了人们对它进行深入研究，邻域设计、多初始点选择、邻域搜索方式、停止策略以及有关的复杂性分析均成为研究中的重要内容[Papadimitriou & Steiglitz, 1982; Johnson *et al.*, 1988; Yannakakis, 1990]。

由于局部搜索方法在理论和应用上的重要性，对此技术所做的任何通用改进都是非常有意义的。其中，关于多初始点的选择，常用策略是均匀随机策略，即从搜索空间中按均匀分布独立重复选取多个初始点。1989 年，贝尔实验室的 Wong 和 Morris 提出了一种基于划分搜索空间的选择多初始点的新策略——划分策略，证明了使用同样数目的初始点，在任一个实例下，任给搜索空间的一个划分，基于此划分的划分策略的性能优于基于同一划分的随机初始点策略(random initial point strategy，一种由 Wong 和 Morris 定义的基于划分搜索空间的非均匀随机取样策略)[Wong & Morris, 1989]。1991 年，Morris 和 Wong 又将此方法和相应结果推广至连

续优化问题并允许邻域搜索带有随机性[Morris & Wong, 1991]。

但是，[Wong & Morris, 1989]中的随机初始点策略只有在划分是均匀划分时才等价于常用策略。因此从[Wong & Morris, 1989]仅能推出基于均匀划分的划分策略即均分策略优于常用策略。这就自然引出如何评价非均匀划分策略以及什么是最好的划分策略的问题。Wong & Morris [1989]把这个问题作为遗留问题提出来而没有解决。此外，划分策略如果比常用策略有优势，那么这种优势有多大？划分策略的实质是什么？对此Wong & Morris [1989]没有给出理论分析。

本文对以上问题做了解答，从而对划分策略的性能做出了完整的评价。我们在第 2 节定义了划分类的均匀性偏序关系，定义了与划分策略有关的各种多初始点选择策略，以及必要的概念和符号约定；在第 3 节定义了划分类的平均划分性能和最坏划分性能两个性能判据，证明了划分类越均匀，相应策略的性能就越好，不仅证明了均分策略是最优的划分策略，而且对不同的非均分策略的性能也进行了比较，彻底解决了 Wong & Morris [1989]提出的遗留问题；在第 4 节给出了划分策略的性能上界，给出了作为最优划分策略的均分策略的性能上下界，从而可以准确估计均分策略比常用策略性能提高的界限；在第 5 节分析并指出划分策略的实质仅仅是企图避免常用策略中初始点可能的重复，而这不是改进常用策略的有效途径，因此划分策略不是一个有前途的初始点选择策略；最后，我们总结了本文的工作，并对由初始点问题引出的有关试验设计方法的问题进行了讨论，指出了几个重要的研究问题。

§2. 概念和符号

本节介绍必要的概念和约定，并尽可能采用与[Wong & Morris, 1989]一致的术语和符号以便于比较。

2.1. 集合的划分，划分类及均匀性偏序关系

集合 S 的子集族 $P = \{S_1, S_2, \ldots, S_m\}$若满足 $\forall 1 \le i \ne j \le m$， $S_i \subseteq S$，

$S_i \neq \phi$，$S_i \cap S_j = \phi$，$\bigcup_{i=1}^{m} S_i = S$，则称 P 为 S 的一个 m 划分，S_i 称为 P 的一个划分块。

有限集合 S 的一个 m 划分 P，若其中大小为 k_i 的划分块分别有 α_i 个，满足 $\forall 1 \le i \ne j \le t$，$k_i, \alpha_i \in \mathbf{N}$，$k_i \ne k_j$，$\sum_{i=1}^{t} \alpha_i = m$，$\sum_{i=1}^{t} k_i \alpha_i = |S|$，则称划分 P 的类型为 $T=k_1^{\alpha_1} k_2^{\alpha_2} \cdots k_t^{\alpha_t}$。类型 T 同时也表示所有类型为 T 的 S 的划分的集合，称作 S 的一个划分类。

设 T 是 S 的一个 m 划分类，T 中任一划分的划分块大小分别为 $k_1, k_2, \cdots, k_m$，若 $\exists i, j$，$1 \le i \ne j \le m$，$k_i \le k_j - 2$，则令 $k_l' = k_l$，$1 \le l \le m$，$l \ne i, j$；$k_i' = k_i + 1$，$k_j' = k_j - 1$。设划分块大小分别为 $k_1', k_2', \cdots, k_m'$ 的划分所在的划分类为 T'，我们定义划分类 T 的均匀性小于划分类 T' 的均匀性，记作 $T < T'$。将 S 的 m 划分类间这个均匀性关系“<”扩充为它的自反传递闭包，记为“≤”。易证“≤”构成 S 的 m 划分类间关于均匀性的一个偏序关系。从定义过程不难看出这个偏序关系的定义与我们对划分类均匀性关系的直观认识是一致的。

引理 1： 给定有限集合 S，$|S| = s$，给定划分数 $m \in \mathrm{N}$，$2 \le m \le s$，且有 $s = qm + r, 0 \le r < m$。在如上定义的 S 的 m 划分类间的均匀性偏序关系“≤”下，存在最大元。当 $r=0$ 时，最大元为 q^m；当 $0<r<m$ 时，最大元为 $q^{m-r}(q+1)^r$。

证明： 设 T 为 S 的任一个 m 划分类，其中任一划分的划分块大小分别为 $k_i, 1 \le i \le m$。

(1) $r = 0$

若 $T = q^m$，则显然有 $T \le q^m$。

若 $T \ne q^m$，则必 $\exists i, 1 \le i \le m, k_i \ne q$，不妨设 $k_1 \ne q$。若 $k_1 > q$，则必存在 $1 \le j \le m, k_j < q$（否则 $\sum_{i=1}^{m} k_i > s$）。不妨设 $k_2 < q$，从而有 $k_2 \le k_1 - 2$。令 $k_1' = k_1 - 1$，$k_2' = k_2 + 1$，$k_i' = k_i, 3 \le i \le m$，设划分块大小分别为 $k_i', 1 \le i \le m$ 的划分所在的划分类为 T_1，则依定义有 $T \le T_1$。若 $k_1 < q$，同理可构造 T_1，使 $T \le T_1$。

若 $T_1 = q^m$，则 $T \le q^m$，证毕。否则可构造 T_2，$T_1 \le T_2$。依此类推，或者 $\exists n$，使 $T \le T_1 \le T_2 \le \cdots \le T_n = q^m$，从而 $T \le q^m$；或者得到一个无限上升链 $T \le T_1 \le T_2 \le \cdots$，而这是不可能的，因为 S 的 m 划分类是有限的。

因此，r=0 时，S 的任一 m 划分类 $T \le q^m$。从而 q^m 是最大元。

(2) $0 < r < m$

若 $\forall 1 \le i \ne j \le m$，$|k_i - k_j| \le 1$，设 $k_i = p$，$1 \le i \le n_1$；$k_j = p+1$，$n_1 + 1 \le j \le m$，其中 $0 < n_1 < m$，则 $T = p^{n_1}(p+1)^{m-n_1}$，由 $\sum_{i=1}^{m} k_i = s$ 知有 $s = n_1 p + (m-n_1)(p+1) = pm + (m-n_1)$，由带余除法唯一性知 $p=q$，$m - n_1 = r$，从而 $T = q^{m-r}(q+1)^r$。

下设 $T \ne q^{m-r}(q+1)^r$，则必 $\exists i, j, |k_i - k_j| \ge 2$。与 r=0 情况类似，可构造 T_1，使 $T \le T_1$。并进一步有 $\exists n, T \le T_1 \le T_2 \le \cdots \le T_n = q^{m-r}(q+1)^r$。

因此 0<r<m 时，S 的任一 m 划分类 $T \le q^{m-r}(q+1)^r$。 ∎

当 $|S| = qm$ 时，我们称划分类 q^m 及其中的任一个划分为均分。由于 $|S|$ 不一定可被 m 除尽，因此 S 的 m 划分类中最大元不一定能达到均分。

2.2. 几种多初始点选择策略

不失一般性，我们以最小化问题作为研究对象。给定一个最小化问题的实例（S，f），其中 S 是离散、有限的可行点集即搜索空间，$f: S \to \mathrm{R}$ 是目标函数，我们要求 f 在 S 上的最小值。

给定一个邻域搜索算法 A，本文要求 A 是确定的，即从同一实例的同一初始点出发所经过的路径是唯一的，此外对 A 没有任何其他限制；给定一个初始点选择策略 I，按照 I 从 S 中选择 n 个点 $x_1, x_2, \cdots, x_n$（可能有重复）作为初始点，设从 x_i 出发经算法 A 所给出的局部最小值为 $A(x_i)$，则称 $B_n^I = \min\{A(x_i), 1 \le i \le n\}$ 为策略 I 下 n 个初始点 $x_1, x_2, \cdots, x_n$ 经算法 A 局部搜索求出的实例（S，f）的解。一般地讲，B_n^I 只是近似最小值；当初始点选择策略 I 带有随机性时，B_n^I 只能以一定概率达到精确最小值，而比较不同初始点选择策略的性能，就是比较它们在其它条件相同时相应的 B_n^I 达到精确最小值的概率。为估计此概率，我们要用到一个“吸引域”

的概念，定义为 $D(r)=\{x\in S\,|\,A(x)>r\}$，其中 $r\in \mathrm{R}$。以上约定均与[Wong & Morris, 1989]一致。

下面介绍几种初始点选择策略。设初始点个数为 n，划分数为 m，为方便说明与计算，不失一般性，假设 $n=km, k\geq 1$：

- 均匀随机初始点选择策略 R_0

对 S 进行 n 次有放回的均匀独立取样，得 n 个初始点，相对应的解记为 $B_n^{R_0}$。这是我们所说的常用策略。

- 基于划分 $P=\{S_1,S_2,\cdots,S_m\}$ 的初始点选择策略 P

在 S_i 中均匀随机选择 k 个点，组成 $mk=n$ 个初始点，相对应的解记为 B_n^P。这是[Wong & Morris, 1989]提出的新策略——划分策略。

- 基于划分 $P=\{S_1,S_2,\cdots,S_m\}$ 的非均匀随机初始点选择策略 RP

以 $1/m$ 的概率选中 $\{S_1,S_2,\cdots,S_m\}$ 中的一个划分块 S_i，再从 S_i 中均匀随机取一点，独立重复此过程 n 次得到 n 个初始点，相对应的解记为 B_n^{RP}。Wong & Morris [1989]称此策略为 “random initial point strategy”，与 R_0 策略是不同的。

- 基于随机选择划分类 T 中划分的初始点选择策略 T

在 T 中随机选择一个元素 P，再以基于划分 P 的初始点选择策略选 n 个初始点，相对应的解记为 B_n^T。

- 均匀随机无放回取样策略 $R(m,k)$

从 S 中均匀无放回取样 m 个点，独立重复此过程 k 次，得 $mk=n$ 个初始点，相对应的解记为 $B_n^{R(m,k)}$。

[Wong & Morris, 1989]的成果是证明了任意实例下，任给一个划分 P，有 $\mathrm{P_r}(B_n^P\leq r)\geq \mathrm{P_r}(B_n^{RP}\leq r), \forall r\in \mathrm{R}$。但是常用策略是 R_0 而不是 RP。因此本文将着重比较 P 策略与 R_0、T、$R(m,k)$ 等策略的关系，以深入理解划分策略的性能和实质。

对于与 m 划分有关的策略，我们称 $n=m$ 时的多初始点搜索求解为基于相应策略的一遍试验。同时为了便于比较，也称 $n=m$ 时基于其它策略的多初始点搜索求解为一遍试验。$n=km$ 时相应称为 k 遍试验。这也同[Wong & Morris, 1989]一致。

§3. 不同划分策略的性能比较

上节我们定义了 m 划分类的均匀性偏序关系，本节我们讨论划分类的均匀性与相应的划分策略的性能之间的关系。这要求我们首先明确以什么样的标准来度量和比较性能。

我们不采用比较两个具体划分的性能的方法，因为任一个划分均可以找到对其极为有利的实例，也可以找到对其极为不利的实例，从而这种比较方法无法获得稳定一致的结论和明确的认识。我们也不采用比较具体划分对所有实例的平均性能或对最坏实例的性能的方法，因为这要求对实例分布、具体的邻域搜索算法决定的吸引域 $D(r)$ 的特点等有更多的了解或假设，这样得出的结论相对比较弱。

我们的方法是比较任一实例下整个划分类的性能，具体地讲，有两个性能标准：一个是划分类中全体划分的平均性能即策略 T 的性能，另一个是划分类中最坏划分的性能。一般地讲，我们并不知道划分类中哪个划分最适合当前实例，选择划分时具有盲目性或随机性，因此从这个角度讲我们比较随机选择划分下的期望性能即划分类中全体划分的平均性能是合理的。这是我们定义第一种性能标准的理由之一。另一个理由请参见定理 1 后面的讨论。第二种标准是为了衡量划分类的最大风险。这两个标准不仅可以获得稳定一致的结论，而且与实例分布、搜索方法等完全无关。

定理 1：任给一个最小化问题的实例 $(S,f),|S|=s$，任给确定的邻域搜索算法 A，任给划分数 $m\in\mathrm{N},2\le m\le s$，任给试验遍数 $k\in\mathrm{N}$，任给性能指标 $r\in\mathrm{R}$，设 $|D(r)|=d$。设 T 是 S 的任一个 m 划分类，$P=\{S_1^0,S_2^0,\cdots,S_m^0\}$ 是 T 中任一个划分，$\left|S_i^0\right|=s_i,1\le i\le m$。则有

$$\mathrm{P_r}(B_{mk}^T\le r)\;=\;1-\frac{\displaystyle\sum_{\forall\{S_1,S_2,\cdots,S_m\}\in T}\left(\prod_{i=1}^{m}\frac{\left|D(r)\cap S_i\right|}{\left|S_i\right|}\right)^k}{|T|}$$

$$=1-\frac{\sum\limits_{\forall D\subseteq S,|D|=d}\left(\prod\limits_{i=1}^{m}\frac{|D\cap S_i^0|}{|S_i^0|}\right)^k}{\binom{s}{d}}$$

$$=1-\frac{\sum\limits_{\forall i_1,i_2,\cdots,i_m,\sum\limits_{1\le j\le m}i_j=d}\left(\prod\limits_{j=1}^{m}\binom{s_j}{i_j}\right)\left(\prod\limits_{j=1}^{m}\frac{i_j}{s_j}\right)^k}{\binom{s}{d}}$$

证明： 只证 $\dfrac{\sum\limits_{\forall\{S_1,S_2,\cdots,S_m\}\in T}\left(\prod\limits_{i=1}^{m}\frac{|D(r)\cap S_i|}{|S_i|}\right)^k}{|T|}=\dfrac{\sum\limits_{\forall D\subseteq S,|D|=d}\left(\prod\limits_{i=1}^{m}\frac{|D\cap S_i^0|}{|S_i^0|}\right)^k}{\binom{s}{d}}$。其余显然。

我们首先证明 $\forall D\subseteq S,|D|=d$，有

$$\sum_{\forall\{S_1,S_2,\cdots,S_m\}\in T}\left(\prod_{i=1}^{m}\frac{|D\cap S_i|}{|S_i|}\right)^k=\sum_{\forall\{S_1,S_2,\cdots,S_m\}\in T}\left(\prod_{i=1}^{m}\frac{|D(r)\cap S_i|}{|S_i|}\right)^k$$

构造 S 上的一一变换 $f:S\to S$，使 $f(D)=D(r)$，则易证 $f_1:T\to T$，$\{S_1,S_2,\cdots,S_m\}\mapsto\{f(S_1),f(S_2),\cdots,f(S_m)\}$ 是 T 上的一一变换。

由 $\dfrac{|D\cap S_i|}{|S_i|}=\dfrac{|f(D\cap S_i)|}{|f(S_i)|}=\dfrac{|f(D)\cap f(S_i)|}{|f(S_i)|}=\dfrac{|D(r)\cap f(S_i)|}{|f(S_i)|}$，从而有

$$\sum_{\forall\{S_1,S_2,\cdots,S_m\}\in T}\left(\prod_{i=1}^{m}\frac{|D\cap S_i|}{|S_i|}\right)^k$$

$$=\sum_{\forall\{S_1,S_2,\cdots,S_m\}\in T}\left(\prod_{i=1}^{m}\frac{|D(r)\cap f(S_i)|}{|f(S_i)|}\right)^k$$

$$=\sum_{\forall\{S_1,S_2,\cdots,S_m\}\in T}\left(\prod_{i=1}^{m}\frac{|D(r)\cap S_i|}{|S_i|}\right)^k$$

同理，$\forall\{S_1,S_2,\cdots,S_m\}\in T$，$|S_i|=s_i$，$1\le i\le m$，构造 S 上一一变换 $g:S\to S$，使 $g(S_i)=S_i^0,1\le i\le m$，同法可证

$$\sum_{\forall D\subseteq S,|D|=d}\left(\prod_{i=1}^{m}\frac{|D\cap S_i|}{|S_i|}\right)^k=\sum_{\forall D\subseteq S,|D|=d}\left(\prod_{i=1}^{m}\frac{|D\cap S_i^0|}{|S_i^0|}\right)^k$$

因此，$\dfrac{\sum_{\forall\{S_1,S_2,\cdots,S_m\}\in T}\left(\prod_{i=1}^{m}\dfrac{|D(r)\cap S_i|}{|S_i|}\right)^k}{|T|}$

$$=\frac{\sum_{\forall D\subseteq S,|D|=d}\ \sum_{\forall\{S_1,S_2,\cdots,S_m\}\in T}\left(\prod_{i=1}^{m}\frac{|D\cap S_i|}{|S_i|}\right)^k}{\binom{s}{d}\cdot|T|}$$

$$=\frac{\sum_{\forall\{S_1,S_2,\cdots,S_m\}\in T}\ \sum_{\forall D\subseteq S,|D|=d}\left(\prod_{i=1}^{m}\frac{|D\cap S_i|}{|S_i|}\right)^k}{|T|\cdot\binom{s}{d}}$$

$$=\frac{\sum_{\forall D\subseteq S,|D|=d}\left(\prod_{i=1}^{m}\frac{|D\cap S_i^0|}{|S_i^0|}\right)^k}{\binom{s}{d}}$$

∎

定理 1 实质上是一个非常一般的公式，在本文具体背景下又有非常特殊的含义。如果假设“若某一实例在 r 下的吸引域 $|D(r)|=d$，则任一大小为 d 的 S 的子集均是某一实例在 r 下的吸引域，且这样的实例分布均匀”成立，那么定理 1 表明基于随机选择划分类中划分的初始点选择策略 T 的算法性能等于基于划分类中任一划分的初始点选择策略 P 的算法对于不同实例的平均性能。但是我们可以保证选择划分的随机性，即策略 T 是可以实现的，却无法保证实例按假设分布。这揭示了随机算法性能与确定算法对实例平均的性能之间的联系与区别。定理 1 也进一步说明我们选择划分类中全体划分的平均性能作为比较标准的合理性，即在保证可实现性而不依赖于假设的同时，在一定意义下也对具体划分的平均性能作出了比较。

推论 1：就一遍试验而言，任一 m 划分类的平均划分性能均等于 $R(m,1)$ 策略的性能。即

$\forall (S,f),|S|=s,\quad \forall A,\quad \forall m\in \mathrm{N},2\le m\le s,\quad \forall r\in \mathrm{R},|D(r)|=d,\quad \forall\quad m$

划分类 T，$\mathrm{P_r}(B_m^T\le r)=\mathrm{P_r}(B_m^{R(m,1)}\le r)=1-\dfrac{\binom{d}{m}}{\binom{s}{m}}$

证明：定理 1 中令 k=1 即得。 ∎

推论 1 提示我们，划分策略与均匀随机策略 R_0 相比，实质上是企图避免 R_0 策略中初始点可能的重复。

引理 2：　$\forall s_1,s_2\in \mathrm{N},s_1\le s_2-2$

$\forall d\in \mathrm{Z},0\le d\le s_1+s_2$

$\forall k\in \mathrm{Z},k\ge 0$，有

$$\sum_{i=0}^{d}\binom{s_1}{i}\binom{s_2}{d-i}\left(\frac{i}{s_1}\cdot\frac{d-i}{s_2}\right)^k\ge\sum_{i=0}^{d}\binom{s_1+1}{i}\binom{s_2-1}{d-i}\left(\frac{i}{s_1+1}\cdot\frac{d-i}{s_2-1}\right)^k$$

证明：见附录。 ∎

定理 2：$\forall (S,f),|S|=s,\forall A,\forall m\in \mathrm{N},2\le m\le s,\forall k\in \mathrm{N},\forall r\in \mathrm{R},|D(r)|=d$，$T$ 是 S 的一个 m划分类，其中任一划分的划分块大小分别为 $s_1,s_2,\cdots,s_m$，且满足 $s_1\le s_2-2$。令 $s_1'=s_1+1,s_2'=s_2-1,s_i'=s_i,3\le i\le m$，设划分块大小分别为 $s_1',s_2',\cdots,s_m'$ 的划分所在划分类为 T'，则有

$$\mathrm{P_r}(B_{mk}^T\le r)\le \mathrm{P_r}(B_{mk}^{T'}\le r)$$

证明：$\mathrm{P_r}(B_{mk}^T\le r)$

$=1-\mathrm{P_r}(B_{mk}^T>r)$

$$=1-\frac{\displaystyle\sum_{\forall i_1,i_2,\cdots,i_m\ge 0,\ \sum_{1\le j\le m} i_j=d}\left(\prod_{j=1}^{m}\binom{s_j}{i_j}\right)\left(\prod_{j=1}^{m}\frac{i_j}{s_j}\right)^k}{\binom{s}{d}}$$

（定理 1）

$$=1-\frac{\sum_{t=0}^{d}\left(\sum_{\forall i_1,i_2\geq 0,i_1+i_2=t}\binom{s_1}{i_1}\binom{s_2}{i_2}\left(\frac{i_1}{s_1}\right)^k\left(\frac{i_2}{s_2}\right)^k\right)\left(\sum_{\forall i_3,\cdots,i_m\geq 0,\sum_{3\leq j\leq m} i_j=d-t}\left(\prod_{j=3}^{m}\binom{s_j}{i_j}\right)\left(\prod_{j=3}^{m}\frac{i_j}{s_j}\right)^k\right)}{\binom{s}{d}}$$

$$\leq 1-\frac{\sum_{t=0}^{d}\left(\sum_{\forall i_1,i_2\geq 0,i_1+i_2=t}\binom{s_1+1}{i_1}\binom{s_2-1}{i_2}\left(\frac{i_1}{s_1+1}\right)^k\left(\frac{i_2}{s_2-1}\right)^k\right)\left(\sum_{\forall i_3,\cdots,i_m\geq 0,\sum_{3\leq j\leq m} i_j=d-t}\left(\prod_{j=3}^{m}\binom{s_j}{i_j}\right)\left(\prod_{j=3}^{m}\frac{i_j}{s_j}\right)^k\right)}{\binom{s}{d}}$$

（引理 2）

$$=1-\frac{\sum_{\forall i_1,i_2,\cdots,i_m\geq 0,\sum_{1\leq j\leq m} i_j=d}\binom{s_1+1}{i_1}\binom{s_2-1}{i_2}\binom{s_3}{i_3}\cdots\binom{s_m}{i_m}\left(\frac{i_1}{s_1+1}\cdot\frac{i_2}{s_2-1}\cdot\frac{i_3}{s_3}\cdots\cdots\frac{i_m}{s_m}\right)^k}{\binom{s}{d}}$$

$$=1-\mathrm{P_r}(B_{mk}^{T'}>r)$$

$$=\mathrm{P_r}(B_{mk}^{T'}\leq r)$$

∎

定理 2 结合上节关于划分类均匀性的偏序关系定义可得推论：

推论 2：划分类越均匀，划分类的平均划分性能越好。即
$\forall(S,f),\forall A,\forall m\in\mathrm{N},2\leq m\leq|S|,\forall k\in\mathrm{N},\forall r\in\mathrm{R}$，$\forall m$ 划分类 T_1,T_2，若 $T_1\leq T_2$，则 $\mathrm{P_r}(B_{mk}^{T_1}\leq r)\leq\mathrm{P_r}(B_{mk}^{T_2}\leq r)$

∎

以上我们比较了不同划分类在平均划分性能标准下的性能。下面讨论最坏划分性能标准下划分类的性能与其均匀性的关系。

定理 3：$\forall(S,f),\forall A,\forall m\in\mathrm{N},2\leq m\leq|S|,\forall k\in\mathrm{N},\forall r\in\mathrm{R}$，设 T 是 S 的一个 m 划分类，其中任一划分的划分块大小分别为 $s_1,s_2,\cdots,s_m$，且满足 $s_1\leq s_2-2$。令 $s_1'=s_1+1,s_2'=s_2-1,s_i'=s_i,3\leq i\leq m$，设划分块大小分别为 $s_1',s_2',\cdots,s_m'$ 的划分所在划分类为 T'，则有

$$\min_{\forall P\in T}\mathrm{P_r}(B_{mk}^{P}\leq r)\leq\min_{\forall P\in T'}\mathrm{P_r}(B_{mk}^{P}\leq r)$$

证明： 设 $\min\limits_{\forall P\in T'}\mathrm{P_r}(B_{mk}^{P}\le r)=\mathrm{P_r}(B_{mk}^{P_1}\le r)$，其中 $P_1\in T'$，$P_1=\{S_1',S_2',\cdots,S_m'\}$，$|S_i'|=s_i',1\le i\le m$，则 $\mathrm{P_r}(B_{mk}^{P_1}\le r)=1-\left(\prod\limits_{i=1}^{m}\dfrac{|D(r)\cap S_i'|}{|S_i'|}\right)^k$

(1) 若 $D(r)\cap S_1'=S_1'$，则 $S_1'\subseteq D(r)$。任取 $x\in S_1'$，令 $P_2=\{S_1,S_2,\cdots,S_m\}$，其中 $S_1=S_1'-\{x\}$，$S_2=S_2'\cup\{x\}$，$S_i=S_i',3\le i\le m$，则有 $|S_i|=s_i,1\le i\le m$，从而 $P_2\in T$。

$$
\begin{aligned}
&\mathrm{P_r}(B_{mk}^{P_1}\le r)\\
&=1-\left(\frac{|D(r)\cap S_1'|}{|S_1'|}\cdot\frac{|D(r)\cap S_2'|}{|S_2'|}\cdot\prod_{i=3}^{m}\frac{|D(r)\cap S_i'|}{|S_i'|}\right)^k\\
&\ge 1-\left(\frac{|D(r)\cap S_1'|}{|S_1'|}\cdot\frac{|D(r)\cap S_2'|+1}{|S_2'|+1}\cdot\prod_{i=3}^{m}\frac{|D(r)\cap S_i'|}{|S_i'|}\right)^k\\
&=1-\left(\frac{|D(r)\cap S_1|}{|S_1|}\cdot\frac{|D(r)\cap S_2|}{|S_2|}\cdot\prod_{i=3}^{m}\frac{|D(r)\cap S_i|}{|S_i|}\right)^k\\
&=\mathrm{P_r}(B_{mk}^{P_2}\le r)\ge\min_{\forall P\in T}\mathrm{P_r}(B_{mk}^{P}\le r)
\end{aligned}
$$

从而 $\min\limits_{\forall P\in T}\mathrm{P_r}(B_{mk}^{P}\le r)\le\min\limits_{\forall P\in T'}\mathrm{P_r}(B_{mk}^{P}\le r)$

(2) 若 $\exists x\in S_1',x\notin D(r)$ 则令 $P_2=\{S_1,S_2,\cdots,S_m\}$，其中 $S_1=S_1'-\{x\}$，$S_2=S_2'\cup\{x\}$，$S_i=S_i',3\le i\le m$，则 $|S_i|=s_i,1\le i\le m$，$P_2\in T$。由于 $x\notin D(r)$，所以 $|D(r)\cap S_1'|=|D(r)\cap S_1|$，$|D(r)\cap S_2'|=|D(r)\cap S_2|$

$$
\begin{aligned}
&\mathrm{P_r}(B_{mk}^{P_1}\le r)\\
&=1-\left(\frac{|D(r)\cap S_1'|}{|S_1'|}\cdot\frac{|D(r)\cap S_2'|}{|S_2'|}\cdot\prod_{i=3}^{m}\frac{|D(r)\cap S_i'|}{|S_i'|}\right)^k\\
&=1-\left(\frac{|D(r)\cap S_1|}{|S_1|+1}\cdot\frac{|D(r)\cap S_2|}{|S_2|-1}\cdot\prod_{j=3}^{m}\frac{|D(r)\cap S_j|}{|S_j|}\right)^k\\
&\ge 1-\left(\frac{|D(r)\cap S_1|}{|S_1|}\frac{|D(r)\cap S_2|}{|S_2|}\prod_{i=3}^{m}\frac{|D(r)\cap S_i|}{|S_i|}\right)^k\\
&=\mathrm{P_r}(B_{mk}^{P_2}\le r)\ge\min_{\forall P\in T}\mathrm{P_r}(B_{mk}^{P}\le r)
\end{aligned}
$$

从而 $\min\limits_{\forall P\in T}\mathrm{P_r}(B_{mk}^{P}\le r)\le\min\limits_{\forall P\in T'}\mathrm{P_r}(B_{mk}^{P}\le r)$ ∎

推论 3：划分类越均匀，划分类的最坏划分性能越好。即

$\forall (S,f),\forall A,\forall m\in \mathrm{N},2\le m\le |S|,\forall k\in \mathrm{N},\forall r\in \mathrm{R}$，$\forall S$ 的 m 划分类 T_1，T_2，若 $T_1\le T_2$，则 $\min\limits_{\forall P\in T_1}\mathrm{P_r}(B_{mk}^{P}\le r)\le \min\limits_{\forall P\in T_2}\mathrm{P_r}(B_{mk}^{P}\le r)$

∎

划分类的平均划分性能和最坏划分性能之间有密切联系。随着试验遍数 k 的增大，最坏划分性能在平均划分性能中所占的比重越来越大，从而尽管 k=1 时不同划分类的平均划分性能相同，随着 k 的增大，不同划分类的平均划分性能逐渐变得有差异。

至此，我们证明了在本文定义的划分类均匀性偏序关系下，按照划分类的平均划分性能和最坏划分性能标准，均有相同的结论：划分类越均匀，相应的初始点选择策略性能越好。

§4. 划分策略的性能估计

本节我们将估计划分策略的性能上下界，从而可以知道划分策略可能比常用策略性能提高的限度。

定理 4：$\forall (S,f),|S|=s,\forall A,\forall m\in \mathrm{N},2\le m\le s,\forall k\in \mathrm{N},\forall r\in \mathrm{R},|D(r)|=d$，$\forall m$ 划分类 T，$\mathrm{P_r}(B_{mk}^{T}\le r)\le \mathrm{P_r}(B_{mk}^{R(m,k)}\le r)=1-\left(\dfrac{\binom{d}{m}}{\binom{s}{m}}\right)^{k}$

证明：

$$\begin{aligned}&\mathrm{P_r}(B_{mk}^{T}\le r)\\&=1-\mathrm{P_r}(B_{mk}^{T}>r)\\&=1-\frac{\sum\limits_{\forall P\in T}\mathrm{P_r}(B_{mk}^{P}>r)}{|T|}\\&=1-\frac{\sum\limits_{\forall P\in T}\mathrm{P_r}(B_{m}^{P}>r)^{k}}{|T|}\end{aligned}$$

$$\le 1-\left(\frac{\sum_{\forall P\in T}\mathrm{P_r}(B_m^P>r)}{|T|}\right)^k$$

$$=1-\mathrm{P_r}(B_m^{R(m,1)}>r)^k \qquad \text{（推论 1）}$$

$$=1-\mathrm{P_r}(B_{mk}^{R(m,k)}>r)$$

$$=\mathrm{P_r}(B_{mk}^{R(m,k)}\le r)$$

$$=1-\binom{d}{m}^k\Big/\binom{s}{m}^k$$

∎

推论 4：$\forall(S,f),|S|=s,\forall A,\forall m\in \mathrm{N},2\le m\le s,\forall k\in \mathrm{N},\forall r\in \mathrm{R},|D(r)|=d$，$\forall S$ 的 m划分类 T，$\min_{\forall P\in T}\mathrm{P_r}(B_{mk}^P\le r)\le \mathrm{P_r}(B_{mk}^{R(m,k)}\le r)=1-\binom{d}{m}^k\Big/\binom{s}{m}^k$

∎

定理 4 及推论 4 告诉我们任意 m 划分类的平均划分性能及最坏划分性能的上界。这样我们可以知道，如果存在比 R_0 好的划分策略，那么至多能提高多少。为此我们先讨论是否存在比 R_0 好的划分策略。

定理 5：$\forall(S,f),|S|=s,\forall A,\forall m\in \mathrm{N},m|s,\forall k\in \mathrm{N},\forall r\in \mathrm{R},|D(r)|=d$，$\forall S$ 的 m均匀划分 $P=\{S_1,S_2,\cdots,S_m\}$，$|S_i|=\frac{s}{m},1\le i\le m$，有

$$\mathrm{P_r}(B_{mk}^{R_0}\le r)\le \mathrm{P_r}(B_{mk}^P\le r)$$

证明：$\mathrm{P_r}(B_{mk}^P\le r)$

$$=1-\left(\prod_{i=1}^m\frac{|D(r)\cap S_i|}{|S_i|}\right)^k$$

$$\ge 1-\left(\sum_{i=1}^m\frac{|D(r)\cap S_i|}{m\cdot|S_i|}\right)^{m\cdot k}$$

$$=1-\left(\frac{|D(r)|}{s}\right)^{mk}$$

$$=1-\left(\frac{d}{s}\right)^{mk}$$

$$=\mathrm{P_r}(B_{mk}^{R_0}\le r)$$

∎

推论 5：$\forall(S,f),|S|=s,\forall A,\forall m\in \mathrm{N},\forall m\in s,s=qm,\forall k\in \mathrm{N},\forall r\in \mathrm{R},|D(r)|=d$，

设 $T=q^m$，则

$$\mathrm{P_r}(B_{mk}^{R_0}\le r)\le \mathrm{P_r}(B_{mk}^{T}\le r)\le \mathrm{P_r}(B_{mk}^{R(m,k)}\le r)$$

$$\mathrm{P_r}(B_{mk}^{R_0}\le r)\le \min_{\forall P\in T}\mathrm{P_r}(B_{mk}^{P}\le r)\le \mathrm{P_r}(B_{mk}^{R(m,k)}\le r)$$

证明：由定理 4、推论 4、定理 5 直接推出。 ∎

由上可见，任一个均匀划分下的初始点选择策略均优于常用的均匀随机策略 R_0，并进一步保证均匀划分类的性能在平均划分性能和最坏划分能两种标准下均优于常用策略 R_0。一般地讲，所有这些性质是非均匀划分或划分类（即使是最大元）所不具备的，可以构造反例来说明这一点。在实际问题中，均分搜索空间是容易实现的。所以我们可以说存在唯一的一种划分策略即均分策略在任一个实例下均优于常用策略 R_0。由推论 5 我们可以估计这种优势的界限。由于初始点的个数以及划分数相对搜索空间的大小是微不足道的，因此 R_0 策略与 $R(m,k)$ 策略的区别不会很大，从而均分策略对常用策略 R_0 的改进是非常有限的。

§5. 划分策略的实质

经过以上分析，我们对于划分策略的性能有了更为深入、全面的认识。那么，划分策略的实质是什么呢？

设划分数为 m，则选择 mk 个点相当于按给定的初始点选择策略进行 k 遍独立重复试验。我们先讨论一遍试验的特点。任给一个 m 划分，基于此划分的划分策略的第一个特点是每遍试验选出的 m 个初始点互不相同。这个特点与 $R(m,1)$ 策略相近，却与 R_0 策略不同，因为 R_0 策略所选出的点可以有任意多个相重，尽管概率极小。基于给定划分的划分策略的第二个特点是选中空间任一区域的概率是不均匀的，即不仅仅与区域大小有关，而且与区域及各划分块的具体构成有关。而 R_0、$R(m,k)$ 策略选中空间任一区域的概率仅与该区域的大小有关，与区域的具体构成无关。

定理 1 告诉我们，如果从划分类的平均划分效果来看划分策略的性能，则此时选中空间任一区域的概率变得均匀了，即与区域、划分块的组成

没有关系了。这样具体划分下选择空间区域的概率的不均匀性通过对划分类中全体划分平均而消失了。而推论 1 则进一步突出了划分策略的第一个特点，即每遍选择从平均看与 $R(m,1)$ 策略完全相同。而 $R(m,1)$ 策略与 R_0 策略的唯一区别是避免了初始点可能的重复。因此我们说从划分类的平均效果看，划分策略的实质仅仅是企图避免初始点可能的重复。

我们进一步说明 k 遍试验的特点。任给一个实例，一个划分类中各个划分下一遍试验的性能是不均匀或者说有差异的。k 遍重复试验将这一差异进一步放大。所以尽管一遍试验下划分类的平均划分性能等于 $R(m,1)$，k 遍试验之后划分类的平均划分性能就不如 $R(m,k)$ 策略了（见定理 4），甚至不能保证优于 R_0。而且，划分类的均匀性越差，其中各个划分下一遍试验的性能差异越大，经 k 遍试验放大更严重。因此尽管一遍试验下所有 m 划分类的平均划分性能都相同，k 遍试验之后，不同划分类的平均划分性能有了差别，均匀性差的划分类平均划分性能也较差（见定理 2）。所以，k 遍试验之后，划分类中各划分的性能差异或不均匀性成为主要矛盾，划分策略企图避免初始点重复所带来的优势逐渐丧失，导致平均划分性能与 $R(m,k)$ 策略性能的差距拉大。

综上所述，我们认为划分策略的实质仅仅是企图避免初始点可能的重复。但达到这一目的存在比划分策略更有效的策略，即 $R(m,k)$ 策略。$R(m,k)$ 策略无论从简单性、稳定性还是性能上均明显优于划分策略，也稳定地优于常用策略 R_0。由于我们很难认为 $R(m,k)$ 策略显著优于常用策略 R_0，我们更可以说划分策略不是一个有前途的新策略。

§6. 总结和讨论

本文针对[Wong & Morris, 1989]的工作及有关问题做了比较完整深入的探讨，对于不同划分策略的优劣及性能估计给出了明确的回答。证明的结论对任一个实例都有效，不必假设实例的分布，而且在随机比较的意义上已经达到最强。我们认为，划分策略的本质仅仅是企图避免初始点可能的重复，而这不是提高初始点策略性能的一条有前途的路。尽管

我们的证明是在离散搜索空间和确定性邻域搜索方式下做出的，我们认为在连续搜索空间和随机性邻域搜索方式下结论是相同的。

一个进一步的问题是，能否设计一个比均匀随机无放回策略更好的初始点策略呢？局部搜索方法的初始点问题实际上是一个试验设计问题，因为可以把从每个初始点开始的局部搜索看作是在这个初始点上进行的一次试验。试验设计是数理统计的一个重要分支，其中有一些重要的试验设计方法，如正交设计法[项可风 & 吴启光, 1989]，均匀设计法[方开泰 & 王元, 1996]，在科研和生产实践中取得了很好的效果。因此，我们想应用这些试验设计方法来设计局部搜索方法的初始点，期望能取得比均匀随机无放回策略更好的性能。

初步研究表明，在$\{0, 1\}^n$的离散空间中无法定义点集的偏差(discrepancy)，因而均匀设计无法应用，而构造任意大小的正交点集也存在困难。由于正交设计和均匀设计都是按某种均匀性度量设法把点分布得均匀一些，因此我们进一步考虑在$\{0, 1\}^n$中引入一种点集的均匀性度量，然后以此为目标函数，用局部搜索方法对均匀随机无放回策略产生的点集进行优化，期望取得更好的性能。但是，我们发现很难在任一个实例上均保证某种方式生成的点集比均匀随机无放回策略产生的点集性能好，即使在一些较弱的意义下。我们回过头来研究正交设计和均匀设计，发现这两种方法都建立在点集的某种均匀性越好，对目标函数优化的性能越好的假设上，这个假设仅仅是个直观认识，在数学上并没有严格证明。我们怀疑可能难以建立这种关系，使之对任意实例都有效。这可能就是正交设计和均匀设计均可以给对方找到反例的原因[张里千, 1995; 刘婉如 *et al.*, 1995; 周芝英 *et al.*, 1995; 李久坤, 1996]。这个问题有待数学上进一步研究。

另一个重要的问题是局部搜索方法和试验设计方法的关系问题。这两种方法都是优化方法，一般地讲，局部搜索方法适用于每个点上的目标函数计算较快，搜索空间维数较高的问题，特点是计算密集(computation intensive)；而试验设计方法适用于每个点上的目标函数计算代价较大，搜索空间维数不太高的问题，特点是对试验次数限制较大。这两种方法

都有相当的理论深度，同时都在实用中经受了考验。对局部搜索方法而言，试验设计方法最可能的应用是局部搜索算法参数的调优问题。局部搜索算法在每组参数值下的计算非常费时，因此希望用尽可能少的调节次数达到一个满意的参数配置；同时参数个数也不过几个，因此正适合用试验设计方法来指导算法参数的调优。对试验设计方法而言，局部搜索方法可以帮助寻找试验设计所要求的点集。目前，不仅试验设计涉及点集构造，一些重要的高技术问题中也涉及点集构造，$\{0, 1\}^n$ 中的点集就有非常重要的应用。现在的方法倾向于从数学上直接构造特定点集，难度非常大，效果也不一定好。用局部搜索方法既可以按点集的某种性质为目标函数寻找合适的点集，而且可以对直接构造出的点集进一步优化，技术上可能更现实一些，适用面也更广一些。这是一个重要的研究方向。

§7. 附录

引理 2： $\forall s_1, s_2 \in \mathrm{N}, s_1 \le s_2 - 2$

$\forall d \in \mathrm{Z}, 0 \le d \le s_1 + s_2$

$\forall k \in \mathrm{Z}, k \ge 0$，有

$$\sum_{i=0}^{d}\binom{s_1}{i}\binom{s_2}{d-i}\left(\frac{i}{s_1}\cdot\frac{d-i}{s_2}\right)^k \ge \sum_{i=0}^{d}\binom{s_1+1}{i}\binom{s_2-1}{d-i}\left(\frac{i}{s_1+1}\cdot\frac{d-i}{s_2-1}\right)^k$$

证明：（卢旭光）

1。d=0，d=1 显然成立。下设 $d \ge 2$

2。证明 $s_1 = 1$ 时不等式成立。用数学归纳法证明，对 k 归纳

k=0,1 显然成立

设 $\sum_{i=0}^{d}\binom{s_1}{i}\binom{s_2}{d-i}\left(\frac{i}{s_1}\cdot\frac{d-i}{s_2}\right)^k \ge \sum_{i=0}^{d}\binom{s_1+1}{i}\binom{s_2-1}{d-i}\left(\frac{i}{s_1+1}\cdot\frac{d-i}{s_2-1}\right)^k$

即 $\binom{s_2}{d-1}\left(\frac{d-1}{s_2}\right)^k \ge \binom{2}{1}\binom{s_2-1}{d-1}\left(\frac{d-1}{2(s_2-1)}\right)^k + \binom{2}{2}\binom{s_2-1}{d-2}\left(\frac{d-2}{s_2-1}\right)^k$

由 $\frac{d-1}{s_2}>\frac{d-1}{2(s_2-1)}$，$\frac{d-1}{s_2}\geq\frac{d-2}{s_2-1}$，从而有

$$\sum_{i=0}^{d}\binom{s_1}{i}\binom{s_2}{d-i}\left(\frac{i}{s_1}\cdot\frac{d-i}{s_2}\right)^{k+1}$$

$$=\binom{s_2}{d-1}\left(\frac{d-1}{s_2}\right)^{k+1}$$

$$\geq\binom{2}{1}\binom{s_2-1}{d-1}\left(\frac{d-1}{2(s_2-1)}\right)^{k+1}+\binom{2}{2}\binom{s_2-1}{d-2}\left(\frac{d-2}{s_2-1}\right)^{k+1}$$

$$=\sum_{i=0}^{d}\binom{s_1+1}{i}\binom{s_2-1}{d-i}\left(\frac{i}{s_1+1}\cdot\frac{d-i}{s_2-1}\right)^{k+1}$$

证毕。

3。证明　　$\forall s_1,s_2\in\mathrm{N},\quad 2\leq s_1\leq s_2-2,$

$\forall d\in\mathrm{Z},\quad 2\leq d\leq s_1+s_2,$

$\forall k\in\mathrm{Z},\quad k\geq 0,$

命题成立。用数学归纳法，对 k 归纳。

$k=0,1$ 显然成立

设

$\sum_{i=0}^{d}\binom{s_1}{i}\binom{s_2}{d-i}\left(\frac{i}{s_1}\cdot\frac{d-i}{s_2}\right)^{j}\geq\sum_{i=0}^{d}\binom{s_1+1}{i}\binom{s_2-1}{d-i}\left(\frac{i}{s_1+1}\cdot\frac{d-i}{s_2-1}\right)^{j}$，　$0\leq j\leq k$　成立，

则 $\sum_{i=0}^{d}\binom{s_1}{i}\binom{s_2}{d-i}\left(\frac{i}{s_1}\cdot\frac{d-i}{s_2}\right)^{k+1}$

$$=\sum_{i=1}^{d-1}\binom{s_1-1}{i-1}\binom{s_2-1}{d-1-i}\left(\frac{i}{s_1}\cdot\frac{d-i}{s_2}\right)^{k}$$

$$=\sum_{i=0}^{d-2}\binom{s_1-1}{i}\binom{s_2-1}{d-2-i}\left(\frac{i+1}{s_1}\cdot\frac{d-1-i}{s_2}\right)^{k}$$

$$=\sum_{i=0}^{d-2}\binom{s_1-1}{i}\binom{s_2-1}{d-2-i}\left(\frac{i(d-2-i)}{s_1\cdot s_2}\cdot\frac{(s_1-1)(s_2-1)}{(s_1-1)(s_2-1)}+\frac{d-1}{s_1\cdot s_2}\right)^{k}$$

$$=\sum_{i=0}^{d-2}\binom{s_1-1}{i}\binom{s_2-1}{d-2-i}\sum_{j=0}^{k}\binom{k}{j}\left(\frac{i(d-2-i)}{(s_1-1)(s_2-1)}\right)^{j}\left(\frac{(s_1-1)(s_2-1)}{s_1\cdot s_2}\right)^{j}\left(\frac{d-1}{s_1\cdot s_2}\right)^{k-j}$$

=

$$\sum_{j=0}^{k}\left(\sum_{i=0}^{d-2}\binom{s_1-1}{i}\binom{s_2-1}{d-2-i}\left(\frac{i(d-2-i)}{(s_1-1)(s_2-1)}\right)^j\right)\cdot\binom{k}{j}\left(\frac{(s_1-1)(s_2-1)}{s_1\cdot s_2}\right)^j\left(\frac{d-1}{s_1\cdot s_2}\right)^{k-j}$$

$$\geq\sum_{j=0}^{k}\left(\sum_{i=0}^{d-2}\binom{s_1}{i}\binom{s_2-2}{d-2-i}\left(\frac{i(d-2-i)}{s_1(s_2-2)}\right)^j\right)\cdot\binom{k}{j}\left(\frac{(s_1-1)(s_2-1)}{s_1\cdot s_2}\right)^j\left(\frac{d-1}{s_1\cdot s_2}\right)^{k-j}$$

$$=\sum_{i=0}^{d-2}\binom{s_1}{i}\binom{s_2-2}{d-2-i}\left(\frac{i(d-2-i)}{s_1\cdot(s_2-2)}\cdot\frac{(s_1-1)(s_2-1)}{s_1\cdot s_2}+\frac{d-1}{s_1\cdot s_2}\right)^k$$

=

$$\sum_{i=0}^{d-2}\binom{s_1+1}{i+1}\binom{s_2-1}{d-1-i}\left(\frac{i+1}{s_1+1}\cdot\frac{d-1-i}{s_2-1}\right)\left(\frac{i(d-2-i)(s_1-1)(s_2-1)}{s_1\cdot(s_2-2)\cdot s_1\cdot s_2}+\frac{d-1}{s_1\cdot s_2}\right)^k$$

$$=\sum_{i=1}^{d-1}\binom{s_1+1}{i}\binom{s_2-1}{d-i}\left(\frac{i}{s_1+1}\cdot\frac{d-i}{s_2-1}\right)\left(\frac{(i-1)(d-1-i)(s_1-1)(s_2-1)}{s_1\cdot(s_2-2)\cdot s_1\cdot s_2}+\frac{d-1}{s_1\cdot s_2}\right)^k$$

≡A

若证明 $\dfrac{(i-1)(d-1-i)(s_1-1)(s_2-1)}{s_1\cdot(s_2-2)\cdot s_1\cdot s_2}+\dfrac{d-1}{s_1\cdot s_2}\geq\dfrac{i}{s_1+1}\cdot\dfrac{d-i}{s_2-1}$

对 $1\leq i\leq d-1$ 且 $i\leq s_1+1$ 且 $d-i\leq s_2-1$ 成立

则有 $\mathrm{A}\geq\sum_{i=1}^{d-1}\binom{s_1+1}{i}\binom{s_2-1}{d-i}\left(\dfrac{i}{s_1+1}\cdot\dfrac{d-i}{s_2-1}\right)^{k+1}$

$$=\sum_{i=0}^{d}\binom{s_1+1}{i}\binom{s_2-1}{d-i}\left(\frac{i}{s_1+1}\cdot\frac{d-i}{s_2-1}\right)^{k+1}$$

从而证得

$$\sum_{i=0}^{d}\binom{s_1}{i}\binom{s_2}{d-i}\left(\frac{i}{s_1}\cdot\frac{d-i}{s_2}\right)^{k+1}\geq\sum_{i=0}^{d}\binom{s_1+1}{i}\binom{s_2-1}{d-i}\left(\frac{i}{s_1+1}\cdot\frac{d-i}{s_2-1}\right)^{k+1}$$

证毕。

下面证明　$1\leq i\leq d-1$，$i\leq s_1+1$，$d-i\leq s_2-1$ 时

$$\frac{(i-1)(d-1-i)(s_1-1)(s_2-1)}{s_1\cdot(s_2-2)\cdot s_1\cdot s_2}+\frac{d-1}{s_1\cdot s_2}\geq\frac{i}{s_1+1}\cdot\frac{d-i}{s_2-1}$$

即证　$f(i)\equiv i^2-i\cdot d+\dfrac{(d-1)(s_1+1)(s_2-1)}{s_1+s_2-1}\geq 0$

其中 i 满足　$\max(1,d+1-s_2)\leq i\leq\min(d-1,s_1+1)$

d,s_1,s_2 满足　$2\leq s_1\leq s_2-2,\quad 2\leq d\leq s_1+s_2$

(1) $2 \le d \le 2s_1 + 2$ 时

$$f(i) \ge \min_{\forall i \in \mathrm{R}} f(i) = f\left(\frac{d}{2}\right) = -\frac{1}{4}\left(d^2 - 4 \cdot \frac{(d-1)(s_1+1)(s_2-1)}{s_1+s_2-1}\right) \equiv g(d)$$

$$g(d) = -\frac{1}{4}\left(d^2 - 4 \cdot \frac{(s_1+1)(s_2-1)}{s_1+s_2-1} \cdot d + 4 \cdot \frac{(s_1+1)(s_2-1)}{s_1+s_2-1}\right)$$

由于　$2 < \dfrac{2(s_1+1)(s_2-1)}{s_1+s_2-1} < 2(s_1+1)$

所以　$g(d) \ge \min\limits_{2 \le d \le 2s_1+2} g(d) \ge \min(g(2), g(2s_1+2))$

而　$g(2) > 0, \quad g(2s_1+2) \ge 0$

从而有　$f(i) \ge g(d) \ge 0$

(2) $2s_1 + 2 \le d \le s_1 + s_2$

由于　$i \le \min(d-1, s_1+1) \le s_1 + 1 < \dfrac{d}{2}$

从而有　$f(i) \ge f(s_1+1) \ge 0$.

从而证得在 i, d, s_1, s_2 满足给定条件下 $f(i) \ge 0$ ∎

第四章　用吴方法求解可满足性问题

§1. 研究背景

可满足性问题(Satisfiability Problem，以下简称 SAT 问题)是计算机科学、人工智能及其应用的核心问题，近几年来有关 SAT 问题的求解算法成为研究的热点，获得了许多新的结果和认识。

SAT 问题来源于自动定理证明，因此自动定理证明的一些方法可以直接用于求解 SAT 问题。比较著名的有两个算法：一个是 DP 算法[Davis & Putnam, 1960; Davis *et al.*, 1962]，以它为核心发展的各种改进算法是目前完备算法中效率最高的；另一个是归结法[Robinson, 1965]，它在一阶谓词逻辑自动定理证明中具有漂亮的性质，自然也可以用于命题逻辑自动定理证明，即用于求解 SAT 问题，特别是其中的归结操作与 SAT 问题的各种算法具有深刻联系。

求解 SAT 问题的另一个思路是变换求解。长期以来，在不同的问题背景和问题形式下发展形成了多种问题求解方法，在各自领域中发挥着巨大作用。变换求解即通过某种变换，利用其他领域中比较有效的问题求解方法和策略实现间接求解。变换求解不仅提供了更为广阔的解题思路，而且有助于揭示各种问题及其求解方法之间的内在联系，从多个角度认识问题的本质。目前研究较多的 SAT 问题的变换求解方法与以下四类问题求解方法有关：人工智能界研究较为深入的约束满足问题(Constraint Satisfaction Problem)求解方法，运筹学界的经典成果线性规划(Linear Programming)和整数规划(Integer Programming)方法，成功求解旅行商问题(Travelling Salesman Problem)的局部搜索(Local Search)方法，以及应用数学界广为研究的最优化(Optimization)方法或称非线性规划(Nonlinear Programming)方法。本文则进一步探索用多项式方程组求解方法求解 SAT 问题的可行性。

变换求解最常用的方法是输入变换，即把原问题的输入“译码”成为

新问题形式下的输入，然后调用新问题形式下已有的算法计算，最后将计算的最终结果“解码”为原问题的解答。尽管两种问题形式下的输入和输出分别对应，但是中间计算过程没有对应，即新问题形式下的计算过程相当于一个“黑箱”，无法用原问题形式下的概念和操作予以表达。我们称这种变换求解不具有“可读性”。

变换求解的另一种方法是可读性变换，即把新问题形式下的方法通过某种办法用原问题形式下的概念和操作予以重新表述，从而利用新问题形式下的方法和策略形成原问题形式下的求解方法。显然，可读性变换具有“可读性”。

可读性变换与输入变换相比，其优越性是明显的。输入变换往往使新形式下的输入变得非常庞大，而且无法完全而简洁地表达原问题的许多特性，计算效率会受到影响；输入变换无法有机结合多种求解方法求解同一问题，因为求解方法一般依赖各自不同的问题表示；输入变换也不利于揭示不同问题以及不同求解方法之间的内在联系。可读性变换则在原问题形式下表达各种问题求解策略，不仅可以充分利用原问题的特性，而且可以有机结合、合理裁剪各种问题求解策略形成更有效的方法，这是非常重要的优点，因为目前看来任何单一方法无法有效求解 SAT 问题。此外，可读性变换还可以帮助我们认识不同问题以及不同求解方法之间的内在联系。以求解皇后问题为例，Selman *et al.*[1992]把问题由二元约束满足问题形式经输入变换变成 SAT 问题形式后用局部搜索算法求解，皇后个数不能超过 100；而直接在原问题形式下用局部搜索算法求解，可解的皇后个数超过百万[Minton *et al.*,1990; Sosic & Gu, 1991]。因此，我们在研究变换求解方法时，一定要重视可读性研究，重视可读性变换方法。

可读性变换的实现通常是借助抽象，即把新问题形式下的方法抽象后表达成一种不依赖该问题表示形式的策略，然后将此策略用原问题形式下的语言予以具体化，从而形成求解原问题的新方法。例如，局部搜索原本用于求解旅行商问题，但将局部搜索抽象成一种不依赖具体搜索空间结构的策略后则可以直接用于 SAT 问题求解，目前局部搜索方法是求

解 SAT 问题的不完备算法中效率最高的[Selman *et al.*, 1994; 黄文奇 & 金人超, 1996; 梁东敏 *et al.*, 1996]；而如果把 SAT 问题形式下的实例转化为旅行商问题的一个实例后再用局部搜索求解则将是非常复杂的。另一个典型例子是约束满足问题求解方法中的一个重要概念“强 k 一致性(Strong k-Consistency)”，由于其表达不依赖约束满足问题常用的二元约束表示，因而可以马上应用到 SAT 问题中，用 SAT 问题语言刻画“强 k 一致性”即“子句集在有界$(k-1)$归结($(k-1)$-Bounded Resolution，即归结式长度不超$(k$-1)的归结)运算下封闭后不出现矛盾”。我们常用有界 2 归结、有界 3 归结做预处理或 DP 算法中的节点操作。

但是可读性变换并不都可以靠抽象来实现，因为许多问题求解方法对输入表示依赖较重，仅靠抽象无法完整把握求解方法的实质。本文给出了一个通过输入变换建立基本操作对应从而实现可读性变换的例子。

仅仅提出变换求解的思想是不够的，计算实验是研究变换求解效率必不可少的一个环节。例如[Kask & Dechter, 1995]，局部搜索方法求解均匀的随机 3-SAT 实例比较有效，但对一种簇形的 3-SAT(Clustered 3-SAT)实例效果并不好；用有界 3 归结做预处理后，局部搜索方法却可以很容易地求解簇形 3-SAT 实例，但对均匀的随机 3-SAT 实例效果反而变差。由此可见，没有计算实验，则难于全面评价算法的求解效率。

总之，在 SAT 问题算法的研究过程中，人们认识到必须探索新思路，同时一定要重视可读性研究和计算实验。

本文在上述原则指导下，探索了用吴方法求解 SAT 问题的新思路。吴方法是一种求解多项式方程组的通用算法，它在平面几何定理机器证明中取得的巨大成功引起了人们对吴方法的广泛关注。一个自然的想法是：能否用吴方法求解 SAT 问题呢？本文第 2 节介绍本文用到的吴方法有关内容；第 3 节介绍用吴方法求解 SAT 问题的各个主要环节，着重说明如何通过合适的输入变换建立吴方法的基本操作与子句间有限制的归结操作的对应，进而用 SAT 问题的语言，结合 SAT 问题的特性，将吴方法改变成一种求解 SAT 问题的新方法，完成了可读性变换，并由此获得一些重要认识；第 4 节介绍有关吴方法求解 SAT 问题的详细实验结果，其中

包括吴方法的最优实现以及吴方法与归结法及 DP 算法的比较；最后，第 5 节总结了全文工作。

§2. 吴方法简介

为了全文完整和后面几节方便地引用概念，本节简单介绍本文用到的吴方法有关内容，材料主要来自文献[石赫, 1994; Kapur & Lakshman, 1992]。关于吴方法的全面介绍，请参见[吴文俊, 1984]等更专门的文献。

2.1. 基本概念

设 K 为任一数域，$K[x_1,x_2,\cdots,x_n]$ 为 K 上以 $x_1,x_2,\cdots,x_n$ 为变元的多项式环，简记为 $K[x]$，K^n 为 K 上的 n 维线性空间。给定 $K[x]$ 中的一组多项式 $PS=\{P_1,P_2,\cdots,P_m\}$，给定 K^n 中的一点 x^0，若 $\forall\, 1\le j\le m$，有 $P_j(x^0)=0$，则称 x^0 是多项式方程组 $PS=0$(即 $P_j=0$，$j=1,2,\cdots,m$)的一个解，也称 x^0 是多项式组 PS 的一个零点。多项式组 PS 的全体零点记为 $Zero(PS)$。求解多项式方程组 $PS=0$ 即确定多项式组 PS 的零点集 $Zero(PS)$。

2.2. 变元定序

吴方法求解多项式方程组的第一步是变元定序。变元序对吴方法的求解效率有重要影响，但目前没有一个一般的规则来确定一个好的变元序，只能根据具体问题特点启发式地确定一个序。

设变元定序后重新命名变元使 $x_1<x_2<\cdots<x_n$，即下标越大的变元序越高。任一多项式 $P(x)$ 中实际出现的下标最大的变元称作 $P(x)$ 的主变元。若 x_i 是 $P(x)$ 的主变元，则多项式 $P(x)$ 可以写成

$$P(x) = I\cdot x_i^{d_i} + x_i\text{的低次项}$$

其中 d_i 是 $P(x)$ 对变元 x_i 的最高幂次，记为 $d_i=deg_{x_i}(P)$；I 为首项系数，通常为 $x_1,x_2,\cdots,x_{i-1}$ 的多项式，称为 $P(x)$ 的初式。

定义多项式间的一个关系：$P\ge Q$　iff　(1) P 的主变元高于 Q 的主变元，或(2) P 和 Q 的主变元相同，设为 x_i，$deg_{x_i}(P)\ge deg_{x_i}(Q)$。又定义：

$P \cong Q$ iff $P \geq Q$且$Q \geq P$； $P > Q$ iff $P \geq Q$但$P \cong Q$不成立。

2.3. 多项式求余

多项式间的求余运算是吴方法中的基本运算。任给两个多项式$F, G \in K[x]$，可做G对F的求余运算。设F的主变元为x_i，初式为$I(x_1, x_2, \cdots, x_{i-1})$，一般有

$$I^s \cdot G(x) = \lambda(x) \cdot F(x) + R(x)，\ deg_{x_i}(R) < deg_{x_i}(F)$$

$R(x)$称为多项式G对F的余式，记为$Rem(G/F)$；s是求余中自然出现的幂次，为非负整数。多项式间的求余运算也叫伪除法。

多项式间求余运算的一个基本性质是

$$Zero(F, G) = Zero(F, G, Rem(G/F))。$$

2.4. 升列，余式公式

升列是吴方法中的基本概念。多项式组$AS = \{A_1, A_2, \cdots, A_r\}$称为升列，若满足(1) AS是三角化的，即$\forall 1 \leq i < j \leq r$，$A_i$的主变元低于$A_j$的主变元；(2) AS中的任意两个多项式A_i、A_j，$i < j$，A_j对A_i已经约化，即$deg_{x_i}(A_j) < deg_{x_i}(A_i)$，其中$x_i$设为$A_i$的主变元。数域$K$中的任一非0常数构成一类特殊的升列，称为矛盾升列。

定义升列之间的一个序关系：设有两个升列$AS = \{A_1, A_2, \cdots, A_k\}$，$AS' = \{A'_1, A'_2, \cdots, A'_{k'}\}$，则$AS > AS'$ iff (1)$\exists j, j \leq k, j \leq k'$，使得$\forall 1 \leq i < j, A_i \cong A'_i$,但是$A_j > A'_j$；或(2)$k' > k$,且$\forall 1 \leq i \leq k, A_i \cong A'_i$。我们称多项式组在此序下的极小升列为基列。

多项式对升列求余的余式公式是吴方法中的基本公式。给定一个升列$AS = \{A_1, A_2, \cdots, A_r\}$，$P(x)$为任一多项式，首先$P$对$A_r$求余得余式$R_{r-1}$，然后$R_{r-1}$对$A_{r-1}$求余得余式$R_{r-2}$，依次下去，$R_i$对$A_i$求余得余式$R_{i-1}$，$i = r-2, r-3, \cdots, 1$，即得余式公式

$$I_1^{s_1} \cdot I_2^{s_2} \cdots I_r^{s_r} \cdot P = Q_r \cdot A_r + Q_{r-1} \cdot A_{r-1} + \cdots + Q_1 \cdot A_1 + R_0$$

其中I_i为A_i的初式，幂次s_i为非负整数，$Q_i(x) \in K[x]$，R_0称为多项式P对升列AS的余式，记为$Rem(P/AS)$。余式R_0一般仍是$K[x]$中的多项式，

满足 $deg_{x_i}(R_0) < deg_{x_i}(A_i), \forall 1 \le i \le r$，其中 x_i 设为 A_i 的主变元。

2.5. 特征列，零点定理

特征列是吴方法中最核心的概念。升列 $CS = \{C_1, C_2, \cdots, C_r\}$ 称为多项式组 $PS = \{P_1, P_2, \cdots, P_m\}$ 的特征列，如果(1) PS 中每一多项式 P_j 对 CS 的余式为 0，即 $\forall 1 \le j \le m, Rem(P_j / CS) = 0$；(2)多项式组 PS 和升列 CS 的零点集满足 $Zero(PS) \subseteq Zero(CS)$。

关于特征列的计算有一基本定理：

引理 1[石赫，1994] 给定多项式组 $PS = \{P_1, P_2, \cdots, P_m\}$，$P_j \in K[x], j = 1,2,\cdots,m$，存在一个机械化算法，经有限步运算之后，可由 PS 得到一基列，使 PS 中每一多项式对此基列求余所得余式均为 0。如果这一基列是非矛盾的，即为多项式组的特征列，记为 CS。若此基列是矛盾的，则多项式方程组 $PS = 0$ 无解。

计算多项式组的特征列一般有两种策略，即宽度优先策略和深度优先策略，后者效率较高[Kapur & Lakshman, 1992]。

由特征列的零点集可以构造性地描述原多项式组的零点集，这就是零点定理：

引理 2[石赫, 1994] (零点结构定理) 设多项式组 $PS = \{P_1, P_2, \cdots, P_m\}$ 的特征列为 $CS = \{C_1, C_2, \cdots, C_r\}$，$C_i$ 的初式为 I_i，$i = 1,2,\cdots,r$，则多项式组 PS 的零点集具有下述结构

$$Zero(PS) = Zero(CS / I) \bigcup \bigcup_{i=1}^{r} Zero(PS, I_i)$$

其中 I 是各初式 I_i 的乘积，$\{PS, I_i\}$ 是把 I_i 添加进 PS 后得到的多项式组，$Zero(CS / I) = Zero(CS) - Zero(I)$。

引理 3[石赫, 1994] (零点分解定理) 给定多项式组 $PS = \{P_1, P_2, \cdots, P_m\}$，其零点集可以分解为一系列特征列的零点集之并

$$Zero(PS) = \bigcup_j Zero(CS_j / I_j)$$

式中指标 j 的集合有限，I_j 是特征列 CS_j 中各多项式的初式的乘积。

§3. 用吴方法求解 SAT 问题

本节介绍用吴方法求解 SAT 问题的各个关键环节，着重说明如何通过合适的输入变换实现可读性变换，如何结合 SAT 问题特性高效实现吴方法求解 SAT 问题的算法。

3.1. 输入变换

用吴方法求解 SAT 问题，输入变换是最自然的思路。这就要设法将 SAT 问题原形式下的输入转化为多项式表示，并在两种形式下的解之间建立对应。

SAT 问题一般指合取范式(Conjunctive Normal Form)可满足性问题，其一般表述为：

给定 n 个布尔变元 $x_1,x_2,\cdots,x_n$，x_i 或 $\bar{x}_i$ 称为变元 x_i 对应的文字，记为 $\tilde{x}_i$，x_i 和 $\bar{x}_i$ 称为关于变元 x_i 的互补文字，$i=1,2,\cdots,n$；给定 m 个子句 $C_1,C_2,\cdots,C_m$，其中 $C_j=\tilde{x}_{j_1}\vee\tilde{x}_{j_2}\vee\cdots\vee\tilde{x}_{j_{k_j}}$，$k_j$ 为子句 C_j 中的文字个数，C_j 中各文字分别对应不同变元，$j=1,2,\cdots,m$。问：是否存在 $x_1,x_2,\cdots,x_n$ 的一组 0,1 赋值，使子句集 $\{C_1,C_2,\cdots,C_m\}$ 中的每个子句在此赋值下为真，即子句集 $\{C_1,C_2,\cdots,C_m\}$ 可满足？

1985 年 Kapur 和 Narendran[1985]提出用基于 Groebner 基的方法进行一阶谓词逻辑自动定理证明，其中自然要讨论命题逻辑自动定理证明，实质上相当于求解 SAT 问题。Groebner 基的方法是一种不同于吴方法的求解多项式方程组的方法，但同样要求先把命题逻辑合式公式变成某种多项式，即进行输入变换。Kapur & Narendran [1985]的方法是把任一合式公式(子句形或非子句形)用布尔连接词“异或”和“与”以及常量 0 和 1 表出，然后把“异或”看成“+”，把“与”看成“×”，从而将合式公式表示成布尔环 $B=(\{0,1\},+,\times)$ 上满足 $x_i^2=x_i, i=1,2,\cdots n$ 的多项式环 $B[x_1,x_2,\cdots,x_n]$ 中的多项式，然后建立解的对应，完成输入变换。虽然[Kapur & Narendran, 1985]是用基于 Groebner 基的方法进行求解与证明的，但同样的输入变换下也可以用吴方法求解。

1992 年白硕[1992]受吴方法启示，探讨了一类有关“知道”的逻辑问题用代数方程表达与求解的可能性，同时给出了一种把 SAT 问题化为多项式方程组求解的输入变换方法。白硕[1992]注意到了[Kapur & Narendran, 1985]的工作，但认为“...把方程定义在布尔环或模 2 的有限域上，而不是定义在有理域或实闭域上...就不容易利用求解有理域或实闭域上多项式方程组的成熟方法...”。因此，白硕[1992]把 SAT 问题转化为有理域或实闭域上的多项式方程组求解问题。具体的输入变换是从子句形式出发，定义转换规则Ψ如下：

(1) $\Psi(\tilde{x}_i)=\begin{cases}1-x_i & \text{如果}\tilde{x}_i=x_i\\ x_i & \text{如果}\tilde{x}_i=\bar{x}_i\end{cases}$

(2) $\Psi(\tilde{x}_{j_1}\vee\tilde{x}_{j_2}\vee\cdots\vee\tilde{x}_{j_{k_j}})=\Psi(\tilde{x}_{j_1})\cdot\Psi(\tilde{x}_{j_2})\cdot\cdots\cdot\Psi(\tilde{x}_{j_{k_j}})$

(3) 子句集$\{C_1,C_2,\cdots,C_m\}$对应标准方程组$\{\Psi(C_j)=0,\ j=1,2,\cdots m\}$

两种形式下的解的对应很简单，表述为：

引理 4[白硕, 1992]　子句集$\{C_1,C_2,\cdots,C_m\}$可满足的充要条件是它的标准方程组$\{\Psi(C_j)=0,\ j=1,2,\cdots m\}$有使未知量取值全为0或1的解。

这种输入变换是很自然的，例如设子句为$C=x\vee\bar{y}\vee z$，则

$C=1$

$\Leftrightarrow x\vee\bar{y}\vee z=1$　　(逻辑方程)

$\Leftrightarrow (x=1)\vee(\bar{y}=1)\vee(z=1)$　　(逻辑方程)

$\Leftrightarrow (x=1)\vee(y=0)\vee(z=1)$　　(逻辑、代数混合方程)

$\Leftrightarrow (1-x=0)\vee(y=0)\vee(1-z=0)$　　(逻辑、代数混合方程)

$\Leftrightarrow (1-x)y(1-z)=0$　　(代数方程)

这样就把子句形式的逻辑方程转化为代数方程，解的对应也很显然。但是[白硕, 1992]没有进一步讨论有关方程组求解等问题。

1996 年孙吉贵 *et al.*[1996]也提出了用吴方法求解 SAT 问题的想法。孙吉贵 *et al.*[1996]的输入变换是从一般的合式公式出发，将逻辑公式转化为整数模 2 的有限域$\mathbf{Z}_2$([孙吉贵 *et al.*, 1996]中记为R_2)上的多项式，并建立解的对应，与[Kapur & Narendran, 1985]的变换思路比较相似。孙吉贵 *et al.*[1996]设想仅计算一个特征列，希望能从中找到解或发现矛盾，或从特征列中的单变元多项式确定相应变元的值以帮助随后进行的局部

搜索等过程，实质上是把吴方法的特征列计算作为一个预处理过程，具有一定启发性。但孙吉贵 *et al.*[1996]对特征列使其初式不为0的零点是否存在的判定不很方便，也没有给出实验结果。

那么，本文采取什么样的输入变换呢？

首先，我们从子句形式出发而不是从一般合式公式出发给出输入变换。这不仅是因为子句形式不失一般性，也是通常讨论的形式，更重要的是子句对应的多项式的计算将非常简单，并使我们能够获得一些关于吴方法求解 SAT 问题的重要认识。3.2、3.3 节将详细讨论这一问题。

其次，从子句形式出发，使用[白硕, 1992]提出的输入变换比较自然，但是其中缺一组重要的约束方程 $x_i(1-x_i)=0$，即 $x_i^2=x_i$，$i=1,2,\cdots,n$。这一组约束方程的加入不仅自然地将解限制在0,1上，使引理 4 不必再判断方程组的解是否具有0,1形式从而更加简洁，更重要的是这一组约束方程具有重要的化简作用，可以避免多项式的计算结果产生任一变元幂次超过一次的项，对算法的有效实现非常重要。3.2、3.3 节将进一步对此进行解释。

第三，用吴方法求解 SAT 问题主要是利用吴方法求解多项式方程组，只用到弱形式的零点定理，即第 2 节介绍的引理 2、引理 3，这只要求把多项式定义在数域上即可，因此把多项式定义在有理数域或整数模 2 的有限域 $\mathbf{Z}_2$ 上均可，不影响本文其余部分的讨论。鉴于 SAT 问题变元为布尔变元的特点，将多项式定义在 $\mathbf{Z}_2$ 上有时会方便一些。但不论定义在什么数域上，约束方程 $x_i(1-x_i)=0$，$i=1,2,\cdots,n$ 均是必不可少的。

因此，本文采取如下的输入变换方法 f：

(1) $f(1)=0, f(0)=1$

(2) $f(\tilde{x}_i)=\begin{cases}1-x_i & \text{如果}\tilde{x}_i=x_i\\ x_i & \text{如果}\tilde{x}_i=\bar{x}_i\end{cases}$

(3) $f(\tilde{x}_{j_1}\vee\tilde{x}_{j_2}\vee\cdots\vee\tilde{x}_{j_{k_j}})= f(\tilde{x}_{j_1})\cdot f(\tilde{x}_{j_2})\cdots f(\tilde{x}_{j_{k_j}})$

(4)子句集 $\{C_1,C_2,\cdots,C_m\}$ 对应的方程组为

$$\begin{cases} f(C_j)=0, j=1,2,\cdots m\\ x_i(1-x_i)=0, i=1,2,\cdots,n\end{cases} \qquad \textbf{(I)}$$

其中 $f(C_j)$ 称为子句多项式，$x_i(1-x_i)$ 称为约束多项式。

定理 1　子句集$\{C_1,C_2,\cdots,C_m\}$的可满足赋值与相应方程组(**I**)的解一一对应。▌

这是一个显然的定理，也可以看作是引理 4 的推论。由于约束方程$x_i(1-x_i)=0$的加入，不必如引理 4 那样再判断方程组的解是否具有0,1形式。

到此为止，输入变换就完成了。这时完全可以把吴方法作为一个黑箱来调用进行求解，只等最终结果即可。这是通常的想法。本文特点是通过输入变换完成可读性变换，以便更深入地认识吴方法求解 SAT 问题的特点和有效地把吴方法与 SAT 问题特性及已有求解经验相结合以求得更好的效率。本文其余部分将深入讨论这些问题。

3.2. 可读性变换

可读性变换能否实现关键是两种问题形式下的基本操作是否存在对应，因此，我们着重研究吴方法求解 SAT 问题时其基本操作多项式间的求余运算有什么新的特点。

定理 2　给定子句C_1,C_2，相应多项式为$f(C_1),f(C_2)$，$f(C_2)$的主变元是x_i。把约束方程$x_k(1-x_k)=0$，$k=1,2,\cdots,n$作为化简条件，则$f(C_1)$对$f(C_2)$求余所得余式有以下 4 种情况：

(1) C_1中不含变元x_i的文字，则

$$Rem(f(C_1)/f(C_2)) = f(C_1)$$

(2) C_1中含变元x_i的文字，但与C_2中所含x_i的文字相同，则

$$Rem(f(C_1)/f(C_2)) = 0$$

(3) C_1与C_2含有关于x_i的互补文字，还含有关于其它变元的互补文字，则

$$Rem(f(C_1)/f(C_2)) = 0$$

(4) C_1与C_2含有关于x_i的互补文字，但不含有关于其它变元的互补文字，则

$$Rem(f(C_1)/f(C_2)) = f(Res(C_1,C_2,x_i))$$

其中$Res(C_1,C_2,x_i)$是子句C_1,C_2关于变元x_i的归结式。

证明：

(1) C_1中不含变元x_i的文字

此时，$deg_{x_i}(f(C_1))=0$，$deg_{x_i}(f(C_2))=1$，

$$f(C_1)=0\cdot f(C_2)+f(C_1)，\ deg_{x_i}(f(C_1))<deg_{x_i}(f(C_2))$$

于是

$$Rem(f(C_1)/f(C_2)) = f(C_1)$$

(2) C_1中含变元x_i的文字，但与C_2中所含x_i的文字相同

设$C_1=C_1'\vee\bar{x}_i$，$C_2=C_2'\vee\bar{x}_i$，C_1'、C_2'中不含x_i的文字，则$f(C_1)=f(C_1')\cdot x_i$，$f(C_2)=f(C_2')\cdot x_i$。

若$C_1'=C_2'\vee C_3'$，则有

$$f(C_1)=f(C_3')\cdot f(C_2)+0$$

否则有

$$f(C_2')\cdot f(C_1)=f(C_1')\cdot f(C_2)+0$$

均有

$$Rem(f(C_1)/f(C_2))=0$$

若$C_1=C_1'\vee x_i$，$C_2=C_2'\vee x_i$，类似可证。

(3) C_1与C_2含有关于x_i的互补文字，还含有关于其它变元的互补文字

设$C_1=C_1'\vee x_i$，$C_2=C_2'\vee\bar{x}_i$，其中C_1'、C_2'不含x_i的文字，但含有关于其它某个变元的互补文字。$f(C_1)=f(C_1')\cdot(1-x_i)$，$f(C_2)=f(C_2')\cdot x_i$，

$$f(C_2')\cdot f(C_1)=-f(C_1')\cdot f(C_2)+f(C_1')\cdot f(C_2')$$

$$deg_{x_i}(f(C_1')\cdot f(C_2'))=0<deg_{x_i}(f(C_2))=1$$

因此

$$Rem(f(C_1)/f(C_2))=f(C_1')\cdot f(C_2')$$

由于C_1'、C_2'仍含有关于某个变元的互补文字，设此变元为x_j，则$f(C_1')\cdot f(C_2')$含有因式$x_j(1-x_j)$，而$x_j(1-x_j)=0$，因此$f(C_1')\cdot f(C_2')=0$，从而

$$Rem(f(C_1)/f(C_2))=0$$

若$C_1=C_1'\vee\bar{x}_i$，$C_2=C_2'\vee x_i$，类似可证。

(4) C_1与C_2含有关于x_i的互补文字，但不含有关于其它变元的互补文字

若 $C_1=x_i$， $C_2=\bar{x}_i$，或 $C_1=\bar{x}_i$， $C_2=x_i$，则

$$Res(C_1,C_2,x_i)=0，\ Rem(f(C_1)/f(C_2))=1$$

这时用子句语言讲就是产生了空子句，用多项式或方程语言讲就是产生了矛盾多项式方程，本文统称为产生矛盾。这时仍有 $Rem(f(C_1)/f(C_2))=f(Res(C_1,C_2,x_i))$。

一般地，设 $C_1=C_0\vee C_1'\vee x_i$， $C_2=C_0\vee C_2'\vee \bar{x}_i$，其中 C_0 是 C_1 与 C_2 的公共部分， C_1'、 C_2' 中不含关于任一变元的互补文字，也不含有相同文字，则 $Res(C_1,C_2,x_i)=C_0\vee C_1'\vee C_2'$。

由 $f(C_1)=f(C_0)\cdot f(C_1')\cdot(1-x_i)$， $f(C_2)=f(C_0)\cdot f(C_2')\cdot x_i$ 可得

$$(f(C_0)\cdot f(C_2'))\cdot f(C_1)=(-f(C_0)\cdot f(C_1'))\cdot f(C_2)+f(C_0)\cdot f(C_0)\cdot f(C_1')\cdot f(C_2')$$

$$deg_{x_i}(f(C_0)\cdot f(C_0)\cdot f(C_1')\cdot f(C_2'))=0<deg_{x_i}(f(C_2))=1$$

从而

$$Rem(f(C_1)/f(C_2))=f(C_0)\cdot f(C_0)\cdot f(C_1')\cdot f(C_2')$$

由 $x_k(1-x_k)=0$， $k=1,2,\cdots,n$ 可知 $x_k{}^2=x_k$， $(1-x_k)^2=1-x_k$，因此 $f(C_0)\cdot f(C_0)=f(C_0)$，从而

$$Rem(f(C_1)/f(C_2))=f(C_0)\cdot f(C_1')\cdot f(C_2')=f(Res(C_1,C_2,x_i))$$

若 $C_1=C_0\vee C_1'\vee \bar{x}_i$， $C_2=C_0\vee C_2'\vee x_i$，类似可证。▌

由定理 2 可知，吴方法求解 SAT 问题时，其基本操作子句多项式间的求余运算之结果仍保持着子句形式，而且前 3 种情况下余式并不代表一个具有新信息的子句，我们称之为平凡的，只有第 4 种情况下余式才对应一个新子句，而且正是两个子句的归结式。因此，吴方法在求解 SAT 问题的整个计算过程中将均是对子句多项式进行运算，而且这种运算对应着子句间一种有限制的归结，其限制性表现在仅对某一子句的主变元进行归结，而不是凡有互补文字就进行归结。

这样，吴方法求解 SAT 问题的过程可以在 SAT 问题的子句表示形式下用子句及其归结操作重新表述，而不必再使用多项式的语言了。在这个意义下，我们称吴方法求解 SAT 问题具有“可读性”。在子句多项式形式下定义的概念和运算等均可以直接定义在子句上，例如子句的主变元、

子句的初式、子句的升列、子句的特征列、子句间的求余运算等等，本文也将不再严格区分子句和子句多项式。这给 SAT 问题的求解带来了全新的概念和方法。

用子句及其归结操作表示吴方法求解 SAT 问题的整个计算过程不仅仅便于理解，而且在实现算法时具有不可替代的优点。子句的表示非常整齐简单，用长度等于变元数、元素为 0,1,–1(分别表示相应变元在子句中不出现，以正文字出现，以负文字出现）的一维数组即可统一表示，进行求余运算或说归结也非常简单。而如果用多项式展开表示子句多项式并进行运算则要复杂得多，效率自然会很差，而实现效率将严重影响对方法的评价。

借助输入变换和定理 1、2 基本上实现了可读性变换，下面我们进一步讨论求解 SAT 问题时吴方法与归结法的联系与区别。这里特别指出一点：本文所说的归结法是指命题逻辑自动定理证明中的归结法，而不是指一阶谓词逻辑自动定理证明中的归结法，前者是后者的特例，二者具有密切联系，但又有很大区别。

归结法在实现中具有较大的不确定性，即使增加各种控制策略如单元子句优先、支持集策略等之后仍然具有较大盲目性，归结过程产生大量对于最终证明没有任何意义的子句，极大地影响了归结法的效率[Wos, 1988]。吴方法在子句形式下的计算不仅仅是一种有限制的归结，吴方法的根本特点在于整个计算过程是围绕特征列的计算展开的，归结操作具有较强的目的性，是一种全新的归结法，与以往的归结法具有根本的不同。特别是对于可满足实例，吴方法一旦计算出某个特征列的使其初式不为 0 的零点存在时，就可以给出一个解从而判定实例可满足而停止计算，不必计算出所有的解；而归结法则必须进行到不能归结出不被已有子句包容的新子句时才可以判定有解而停止计算，这时相当于已将所有的解均明确地表示出来了。本文第 4 节将给出两种方法详细的实验数据，无论实例有解还是无解，吴方法均比归结法效率高得多。

另一方面，1985 年 Haken[1985]证明了对于鸽巢问题(Pigeonhole Problem)，任一归结证明至少包含指数多个不同的子句，这就给出了归结

法求解 SAT 问题的一个指数型下界。吴方法求解 SAT 问题基本上是一个归结过程，特别是对于不可满足实例，只有归结操作（定理 2 情况 4）才有助于导出矛盾，因此吴方法求解 SAT 问题最坏情况下可能受到同样的限制。而且，Haken[1985]的结论和定理 1，2 一起有可能给出吴方法求解一般多项式方程组的一个指数复杂性下界，而一般的复杂性结果多是某种指数型上界，一个非平凡的下界并不容易获得，因此这个结果是有意义的。这里我们不对这一命题严格陈述与证明，因为吴方法计算中的问题分解过程需要专门处理，尽管我们认为这一过程不大可能对复杂性产生根本影响。我们只想说明：通过可读性变换，我们对吴方法求解 SAT 问题以及吴方法本身有了更为深刻的认识，而这些认识在输入变换下是无法获得的。

定理 2 的证明过程中可以明显看到约束方程 $x_i(1-x_i)=0,\ \ i=1,2,\cdots,n$ 的化简作用。在吴方法求解 SAT 问题的整个计算过程中，约束多项式 $x_i(1-x_i),\ \ i=1,2,\cdots,n$ 并不直接进行求余运算，而是隐含在子句多项式的求余运算中起化简作用，用子句语言表述，其作用一是消去重言式（情况 3），二是消去重复文字（情况 4），使余式始终保持简单的子句形式，这样可以提高吴方法在 SAT 问题上的计算效率。这里充分利用了 SAT 问题的特点，3.3 节还将进一步说明这一点。

吴方法求解 SAT 问题尽管可以在子句形式下表述和实现，但是，吴方法是一种全新的归结法，具有以往各种归结策略所完全没有的特点，因此，整个计算过程仍按吴方法的一般步骤展开。下面我们讨论在特征列计算、问题分解和变元定序几个重要环节上如何充分利用 SAT 问题特点，这是可读性变换更为重要的目的和优点所在。

3.3. 特征列计算

特征列的计算是吴方法的核心，自然也是吴方法求解 SAT 问题的核心。下面我们将对特征列的具体计算过程、特征列的形式、特征列使其初式不为0的零点集的判定等问题进行讨论。

特征列的计算采用深度优先策略[Kapur & Lakshman, 1992]，即任一多

项式对基列求余得到非0余式后马上对基列进行修改，而不是等到所有多项式对当前基列求余完成之后再修改基列。后者称为宽度优先策略，一般比深度优先策略效率差。根据 SAT 问题的特点，我们对具体计算过程做了进一步的改进：

在我们定义的输入变换下，要求解的多项式方程组(**I**)包含两类多项式，一类是子句多项式 $f(C_j), j=1,2,\cdots,m$，另一类是约束多项式 $x_i(1-x_i), i=1,2,\cdots,n$。在用深度优先策略计算多项式组 $PS=\{f(C_j)\,|\,j=1,2,\cdots,m\}\cup\{x_i(1-x_i)\,|\,i=1,2,\cdots,n\}$ 的特征列时，我们仅仅处理子句多项式 $f(C_j)$，而且是按照定理 2 计算余式；而对约束多项式 $x_i(1-x_i)$ 不做这种显式处理，而是作为化简条件隐含在定理 2 中。最后若没产生矛盾，则得到一组子句多项式构成的升列 $AS=\{f(C'_k), k=1,2,\cdots,r\}$（其中 $1\le r\le n$），使得 PS 中任一子句多项式 $f(C_j), j=1,2,\cdots,m$ 对升列 AS 按定理 2 求余余式为 0。令

$$CS=AS\cup\{x_t(1-x_t)\,|\,1\le t\le n, \text{且}x_t\text{不是}AS\text{中任一个多项式的主变元}\}$$

重新排序后有 $CS=\{P_1,P_2,\cdots,P_n\}$，其中 P_i 的主变元为 x_i。

定理 3　上述计算过程必定终止。若产生矛盾，则多项式方程组 $PS=0$ 无解；否则，CS 是 PS 的特征列。

证明：

先证终止性。

定理 2 定义的求余运算可以理解为两个多项式按通常吴方法定义的求余运算（可参见第 2.3 节）求出余式后，再依次对约束多项式 $x_i(1-x_i), i=1,2,\cdots,n$ 求余，这时得到的余式即是定理 2 定义的余式。由于任一多项式对约束多项式 $x_i(1-x_i), i=1,2,\cdots,n$ 求余只可能降低而不会升高任一变元的幂次，因此整个计算过程和一般特征列的计算过程一样一定会终止。

再证正确性。

若计算过程产生矛盾，显然有多项式方程组 $PS=0$ 无解。

若计算过程不产生矛盾，我们证明 CS 是 PS 的特征列。

显然，CS 是升列，$Zero(PS)\subseteq Zero(CS)$，$PS$ 中任一约束多项式

$x_i(1-x_i), i=1,2,\cdots,n$ 对 CS 按通常吴方法定义的求余运算求余所得余式为0。我们只需证明 PS 中任一子句多项式 $f(C_j), j=1,2,\cdots,m$ 对 CS 按通常吴方法定义的求余运算求余所得余式也为0。

设 P 为 PS 中任一子句多项式，则已知 P 对升列 AS 按定理 2 求余所得余式为 0。因此 $\exists k, 1\le k\le r$，P 对升列 AS 中子句多项式 $f(C_r'), f(C_{r-1}'), \cdots, f(C_{k+1}')$ 按定理 2 依次求余后所得余式为子句多项式 $R_k\neq 0$，R_k 再对 $f(C_k')$ 按定理 2 求余所得余式 $R_{k-1}=0$。则 R_k 一定是经过定理 2 的情况 1 或 4 而得到，而 R_{k-1} 一定是经过定理 2 的情况 2 或 3 而得到。设 P 对升列 AS 中子句多项式 $f(C_r'), f(C_{r-1}'), \cdots, f(C_{k+1}')$ 按通常吴方法定义的求余运算依次求余后所得余式为一般多项式 R_k'，则 R_k' 的计算过程与 R_k 相比，只是少了在情况 4 中对约束多项式 $x_i(1-x_i), i=1,2,\cdots,n$ 求余这个化简过程，因此 R_k' 和 R_k 相比，所含因式完全相同，只是因式的幂次可能较高。设 $f(C_k')$ 对应 CS 中的 P_h，P 对 CS 中多项式 $P_n, P_{n-1}, \cdots, P_{h+1}$ 按通常吴方法定义的求余运算依次求余后所得余式为 R，则 R 的计算过程与 R_k' 相比，只是可能多了对 $P_n, P_{n-1}, \cdots, P_{h+1}$ 中的约束多项式的求余而把相应变元的因式的幂次降为1，因而 R 和 R_k' 相比，所含因式完全相同，只是有些因式的幂次可能较低。因此，R 和 R_k 相比，所含因式完全相同，只是因式的幂次可能不同。

已知 R_k 对 $f(C_k')$ 即 P_h 按定理 2 求余得0只能由情况 2 或 3 而得到。若是情况 2，则 R_k 与 P_h 均含有关于 x_h 的相同的因式(x_h 或 $1-x_h$)，从而 R 与 P_h 均含有关于 x_h 的相同的因式。由于 P_h 关于 x_h 的因式的幂次为 1，因此 R 对 P_h 按通常吴方法定义的求余运算求余所得余式为0，从而 P 对 CS 按通常吴方法定义的求余运算求余所得余式为0。若是情况 3，则 R 对 P_h 按通常吴方法定义的求余运算求余所得余式中含有因式 $x_g(1-x_g)$(其中 $x_g<x_h$)，由于没有定理 2 中的化简，因而不能马上得0，但是最迟到对 P_g 按通常吴方法定义的求余运算求余时一定为0，从而 P 对 CS 按通常吴方法定义的求余运算求余所得余式为0。

因此，PS 中任一子句多项式 $f(C_j), j=1,2,\cdots,m$ 对 CS 按通常吴方法定义的求余运算求余所得余式也为0。从而证得 CS 是 PS 的特征列。▌

我们仅仅处理子句多项式 $f(C_j)$，而且是按照定理 2 计算余式；对约束多项式 $x_i(1-x_i)$ 不做这种显式处理，而是作为化简条件隐含在定理 2 中，这样子句多项式间的求余运算将具有定理 2 所示的简单结果形式，而不会产生某一变元幂次超过一次的余式，不仅简化了计算过程，而且使我们看到了吴方法求解 SAT 问题与归结法的某种联系。定理 3 则保证了这样处理不失吴方法原来意义下的严格性。

关于特征列使其初式不为0的零点集的判定有下面的简单定理：

定理 4　设 CS 是 PS 的特征列，I 是 CS 中各多项式的初式的乘积，则 $Zero(CS/I)\neq\varnothing$　iff　CS 中各子句多项式的初式对应的子句间不含关于任一变元的互补文字。$Zero(CS/I)\neq\varnothing$ 时，$Zero(CS/I)$ 中的零点可以这样确定：CS 中各子句多项式的主变元对应的子句文字为1(真)，各子句多项式的初式对应的子句的各文字为0(假)，其余变元取值任意。▌

我们称某一变元为冲突变元，如果特征列各子句多项式的初式对应的子句间含有关于此变元的互补文字。这样，特征列使其初式不为0的零点集的判定可以简单表述为：$Zero(CS/I)\neq\varnothing$　iff　特征列的初式中不存在冲突变元。

我们之所以从子句形而非一般合式公式出发对 SAT 问题进行输入变换及应用吴方法求解，正是因为从一般合式公式出发无法有效利用 SAT 问题特点，不仅难于获得吴方法求解 SAT 问题的可读性，进而无法通过吴方法与归结法的比较来更深入地认识吴方法，而且其计算过程将基本上成为一般多项式的计算从而复杂化，最后求出特征列也难于象定理 4 那样简单地判定特征列使其初式不为0的零点是否存在。

特征列是吴方法的精华。尽管吴方法求解 SAT 问题时与归结法有一定联系，但是特征列的概念是吴方法所独有的，这使吴方法成为求解 SAT 问题的一种全新的方法。我们不仅可以按吴方法那样以特征列为中心组织计算，而且可以把特征列计算作为一个子过程嵌入其它算法，使吴方法的成果可以被更好地利用。

3.4. 问题分解

如果特征列的计算没有产生矛盾，但是 $Zero(CS/I)=\varnothing$，其中 $I=I_1\cdot I_2\cdots I_n$，$I_i, i=1,2,\cdots,n$ 是 CS 中各多项式的初式(实际上只有其中的子句多项式的初式有用，约束多项式的初式均为1)，即 CS 的初式中有冲突变元，那末依照引理 2,3 零点定理，就要进一步分别求解多项式组 $\{PS, I_i\}, i=1,2,\cdots,n$。

由于 SAT 问题存在按变元分别取值0或1的自然分解过程，而且单元子句的加入要比长子句的加入有更好的计算效率，因此我们利用这个特点，选择某一变元 x_i，将多项式组 PS 分解为 $\{PS, x_i\}$ 和 $\{PS, 1-x_i\}$ 后分别用吴方法继续求解。在 SAT 问题算法研究中，这一过程称为分支过程，变元 x_i 称为分支变元。在选择分支变元时，我们利用当前特征列的信息，选择 CS 的初式中的冲突变元作为分支变元，目的是尽可能消除或减少新的特征列的初式中出现冲突变元的可能性。当然这也只是一个启发性规则。当有多个冲突变元可选时，有一个选较高序变元还是选较低序变元的选择，分别称为“高冲突变元优先分支”和“低冲突变元优先分支”。在第 4 节我们对这两种分支策略均进行了实验。

问题分解使吴方法求解 SAT 问题的过程可以表示成一棵搜索树，其中每个节点代表一个特征列的计算；叶子节点代表特征列计算发现矛盾，或者特征列使其初式不为0的零点存在；非叶子节点代表特征列计算没有发现矛盾，但是特征列使其初式不为0的零点不存在（即特征列的初式中存在冲突变元）。

3.5. 变元定序

吴方法求解 SAT 问题时，输入变换之后首先要进行变元定序，然后才能开始特征列计算及问题分解等。变元序对吴方法的效率影响很大。

正如 2.2 节所指出的那样，变元定序缺乏一般规则，只能根据问题特点启发式地予以确定。在 SAT 问题的各种算法中也常常涉及对变元排序的问题，一般以变元的约束强弱来排序，而变元的约束强弱也只能启发式地度量。在通常用作实验模型的随机 3-SAT 实例模型下，用变元以正

文字出现的子句数和以负文字出现的子句数之乘积作为变元约束强弱的度量效果较好，乘积越大，约束越强。这样，我们可以获得两种变元序：约束越强，变元序越高，称为“强约束变元高序”；约束越强，变元序越低，称为“强约束变元低序”。此外，为了了进一步检验这两种变元序是否对求解效率有显著影响，我们可以用“随机序”即随机产生的一个变元序作为对比。在第 4 节我们对这 3 种变元序均进行了实验。

§4. 实验及分析

本节介绍有关吴方法求解 SAT 问题的详细实验结果和分析。实验分三部分：第一部分是吴方法的最优实现，主要是通过实验决定采用何种变元序和分支策略最有效；第二部分是吴方法与归结法的比较，因为吴方法求解 SAT 问题与归结法有密切联系；第三部分是吴方法与 DP 算法的比较，因为 DP 算法是目前 SAT 问题完备算法中最有效的。

问题模型采用常用的均匀随机 3-SAT 实例模型[Mitchell *et al.*, 1992]，这是一种易于生成而又非常难解的实例模型，常用于测试各种 SAT 问题算法。设要求生成一个 n 个变元 m 个子句的随机实例，生成方法是：从 n 个变元中随机选 3 个不同的变元，每个变元以 0.5 的概率变反后组成一个 3 个文字的子句，独立重复这一过程 m 次即得。已有实验表明，$m/n \approx 4.3$ 时，生成的实例有 50％可满足，这时实例最难[Mitchell *et al.*, 1992]。

实验用的计算机是 SGI-Elan4000 工作站。所有程序均用 C 语言编写，并经最高级别的优化编译。

4.1. 吴方法的最优实现

在本文定义的可读性变换下，吴方法的实现是比较简单的。子句多项式均按子句形式存放，子句用长度等于变元数、元素为 0,1,–1(分别表示相应变元在子句中不出现，以正文字出现，以负文字出现）的一维数组统一表示；特征列计算采用深度优先策略，仅处理子句多项式，基本运算按定理 2 进行。只有两个环节需要进一步通过实验确定：变元序和分支

策略。我们实验了 3 种变元序：强约束变元高序，强约束变元低序和随机序（参见 3.5 节），2 种分支策略：高冲突变元优先分支和低冲突变元优先分支(参见 3.4 节)，共 6 种组合。实验 1 给出了这 6 种组合对变元数为 50，子句数为 215 的 50 个随机实例（其中 27 个可满足）的实验数据。

实验 1　吴方法实现
变元数=50，子句数=215，实例数=50（其中可满足实例数=27）

		强约束变元低序			随机序			强约束变元高序		
		SAT	UNSAT	TOTAL	SAT	UNSAT	TOTAL	SAT	UNSAT	TOTAL
低冲突变元优先分支	时间	0	1	0	1	6	4	29	109	66
	节点数	17	28	23	36	113	72	328	999	637
	基本运算	35778	70909	51938	121738	421027	259411	1811622	6617467	4022311
	A	0	0	0	0	0	0	0	0	0
	B	0	0	0	0	0	0	0	0	0
	C	0	0	0	0	0	0	0	0	0
高冲突变元优先分支	时间	0	2	1	3	10	6	22	61	40
	节点数	65	141	100	144	433	277	364	904	612
	基本运算	67684	156358	108474	241661	716021	459867	1413003	3813508	2517235
	A	0	0	0	0	0	0	0	0	0
	B	0	0	0	0	0	0	0	0	0
	C	0	0	0	0	0	0	0	0	0

注 a：SAT,UNSAT,TOTAL 分别表示可满足实例，不可满足实例，全体实例的平均数据；
注 b：时间单位为秒，0 表示小于 0.5 秒；
注 c：节点数即计算特征列的次数，参见 3.4 节；
注 d：基本运算是指定理 2 中情况 2,3,4 的总计数；
注 e：A,B,C 为第一个特征列的信息，其中 A 表示第一个特征列有使其初式不为 0 的零点的实例个数，B 表示第一个特征列矛盾的实例个数，C 表示第一个特征列既不矛盾又没有使其初式不为 0 的零点时，特征列中对应单元子句的子句多项式的个数。

从实验 1 的各项数据可以看出：

(1)变元序对算法效率影响非常大。其中，“强约束变元低序”各项指标一致优于“随机序”，“随机序”各项指标一致优于“强约束变元高序”，而且差异是显著的。这对吴方法求解一般问题时的变元定序也有启发，即在定义某种约束后，将约束较强的变元序定得低些可能效果好一些。

(2)分支策略对求解效率影响不如变元序大，相对优劣与变元序有关。分支策略必须配合变元序才能取得较好的效果。

(3)第一个特征列基本上没有可以马上利用的信息。进一步进行问题分

解是必要的。

因此，本文的吴方法实现采用“强约束变元低序”的变元序和“低冲突变元优先分支”的分支策略。我们进一步测试了吴方法在更大规模问题上的性能，发现也是在$m/n \approx 4.3$附近比较难，其余较易。实验 2 给出了变元数为 100，子句数分别为 400、430、500，各 100 个实例的包括有关指标平均、最小、最大值的实验数据。

实验 2　吴方法最优实现下的性能
变元数=100，子句数=400,430,500，实例数各 100

		实例数	时间			节点数			基本运算		
			最小	平均	最大	最小	平均	最大	最小	平均	最大
子句数=400	SAT	99	0	25	171	8	144	857	34546	952470	6335061
	UNSAT	1	125	125	125	591	591	591	4703445	4703445	4703445
	TOTAL	100	0	26	171	8	149	857	34546	989980	6335061
子句数=430	SAT	54	1	38	355	11	165	1133	58457	1394844	12629841
	UNSAT	46	21	83	237	87	313	841	807036	2997091	8661354
	TOTAL	100	1	59	355	11	233	1133	58457	2131877	12629841
子句数=500	SAT	1	4	4	4	15	15	15	150987	150987	150987
	UNSAT	99	8	40	138	21	84	403	305857	1430169	4971739
	TOTAL	100	4	40	138	15	83	403	150987	1417377	4971739

注：数据说明见实验 1 注。

4.2. 吴方法与归结法的比较

3.2 节的分析表明吴方法求解 SAT 问题与归结法有密切联系，因此我们对吴方法和归结法进行了比较。我们选择了支持集归结和语义归结作为比较对象。

支持集归结是 1965 年 Wos 等提出的[Wos *et al.*, 1965]，其思想是：用归结法证明定理$A_1 \wedge A_2 \wedge \cdots \wedge A_m \rightarrow B$实际上是证明$A_1 \wedge A_2 \wedge \cdots \wedge A_m \wedge \overline{B}$不可满足，而矛盾不大可能存在于前提中，因为前提通常是可满足的，因此应尽量避免在$\{A_1, A_2, \cdots, A_m\}$的子句中归结。一般地，子句集$S$的一个子集$T$称为$S$的支持集，如果$S-T$是可满足的。支持集归结就是归结过程中只选取不同时属于$S-T$的子句进行归结。支持集归结是完备的，而且是目前归结定理证明中效率最高的方法[Wos, 1988]，这是我们选择它与吴方法进行比较的原因。

语义归结是 1967 年 Slagle 提出的[Slagle, 1967]，对子句间的归结做

了两点限制，在命题逻辑中表述为：(1)用子句集 S 的一个解释 I (即各命题变元的一组 0,1赋值)将 S 分成两部分，S 中在 I 下为真的子句组成 S_1，S 中在 I 下为假的子句组成 S_2，规定归结只能在 S_1 和 S_2 的子句之间进行，S_1 内或 S_2 内的子句间不允许进行归结；(2)规定命题变元的一个序，S_1 和 S_2 的子句进行归结时，归结变元必须是 S_2 的子句中序最高的变元。语义归结是完备的，支持集归结可以看成是语义归结的特例；此外，语义归结的第 2 个限制与吴方法极为相似，因此我们也把语义归结与吴方法进行了比较。

支持集归结的实现主要需解决支持集的确定问题。定理证明中支持集常由结论取反构成，而 SAT 问题是判定形式，没有结论部分可以利用。我们采用两种方法确定支持集：一种方法是随机生成一个解释，初始子句集中在此解释下为假的子句构成支持集，我们称之为“随机生成解释”；另一种方法是用局部搜索方法寻找一个较好的解释，使初始子句集中在此解释下为假的子句构成的支持集尽可能小，我们称之为“局部搜索解释”。

语义归结的实现需要解决两个问题：划分子句集的解释和变元序。划分子句集的解释的确定我们采用了和确定支持集相同的两种方法，即随机生成一个解释，或由局部搜索方法寻找一个使初始子句集中在此解释下为假的子句尽可能少的解释。变元序的确定我们采用了吴方法实验的 3 种变元序：强约束变元低序，随机序和强约束变元高序。

把局部搜索方法应用到归结法中是一个新的尝试。近几年来，应用局部搜索方法求解 SAT 问题取得了很大进展[Selman *et al.*, 1992, 1994; Gu, 1992, 1993, 1994; 黄文奇 & 金人超, 1996; 梁东敏 *et al.*, 1996]，其特点是：如果实例可满足，则局部搜索方法一般可以非常高效地找到一个解；如果实例不可满足，则局部搜索方法无法给出任何明确的结论，但是一般可以找到一个解释，使在此解释下为假的子句数达到极少。如何利用局部搜索给出的信息加速对不可满足实例的证明一直没有很好的结果。我们把局部搜索用到归结法求解 SAT 问题上，一个目的是在把局部搜索应用到一阶谓词逻辑归结定理证明之前，先在命题逻辑归结定理证明中

实验其可行性；对本文来讲，更重要的目的是提高归结法的效率，尽可能用最好的归结法与吴方法进行比较。

我们实现了以上 6 种语义归结和 2 种支持集归结，以便选出最好的一种与吴方法进行比较。程序实现时，对所有子句均建立索引表，使各变元通过索引表可以方便地访问变元以正文字或负文字出现的子句，从而使归结操作避免不必要的查找；采用单元子句优先归结的策略以提高归结效率；归结生成的子句进行包容检查之后再加入子句集，这一过程虽然最为费时，但如果不做包容检查，则归结法很快就会因生成大量无用子句而陷于瘫痪[Wos, 1988]；子句表最大容量定为 10000 个子句，超过这个限度还没有结果即宣布该归结法失败。实验 3 给出了有关数据。

从实验 3 的各项数据可以看出：

(1)语义归结要比支持集归结好很多。这可能是命题逻辑特有的，也可能与我们的问题模型有关。

(2)语义归结中，变元序的作用非常大，其中，“强约束变元低序”各项指标一致优于“随机序”，“随机序”各项指标一致优于“强约束变元高序”，而且差异是显著的。这与吴方法的实验结果相同。

(3)局部搜索方法的引入对可满足实例的效果是可以预料到的，重要的是它对归结法证明不可满足实例的作用非常显著，使语义归结各项指标改善了 100%，支持集归结各项指标改善了 20%。因此，把局部搜索方法引入一阶谓词逻辑归结定理证明，在子句集的 H 解释空间中进行局部搜索，可能对支持集归结中支持集的选择、语义归结中划分子句集的解释的选择、线性归结中顶子句的选择这些直接影响相应归结方法效率的关键环节有较大帮助，是一个值得研究的有希望的方向。

实验 3　归结法实现 变元数=30，子句数=129，实例数=50（其中可满足实例数=25）									
		语义归结						支持集归结	
		强约束变元低序		随机序		强约束变元高序			
		SAT	UNSAT	SAT	UNSAT	SAT	UNSAT	SAT	UNSAT
随机生成解释	解出的实例数	25	25	9	5	1	0	0	0
	时间	20	20	67	60	100			
	归结操作	23602	23708	63240	51730	101113			
	包容过程调用	20103	20249	56145	44295	87632			
	包容操作	5266310	4932749	17038533	14793143	30461750			
	子句表中子句数	3183	4281	7128	7906	8434			
	初始不满足子句数	17	15	15	13	10			
局部搜索解释	解出的实例数	25	25	25	9	25	0	25	0
	时间	0	8	0	32	0		0	
	归结操作	0	10804	0	27696	0		0	
	包容过程调用	0	9124	0	24139	0		0	
	包容操作	0	1798941	0	5613506	0		0	
	子句表中子句数	129	3104	129	6543	129		129	
	初始不满足子句数	0	1	0	1	0		0	

实验 3(续)　支持集归结：变元数=12，子句数=60，实例数=50(其中可满足实例数=24)								
		解出的实例数	时间	归结操作	包容过程调用	包容操作	子句表中子句数	初始不满足子句数
随机生成解释	SAT	24	1	11227	8441	702798	1889	7
	UNSAT	26	2	12867	9659	945420	2202	7
局部搜索解释	SAT	24	0	0	0	0	60	0
	UNSAT	26	2	9845	7461	700371	1864	1

注 a：表中所给出的数据是归结法在 10000 个子句内得出解答的实例的平均数据；

注 b：归结操作计数中不包括单元归结操作；

注 c：包容过程调用不记单元归结中的包容过程调用；

注 d：包容操作是指包容过程中的基本操作两子句间的包容检查，也不记单元归结中的包容操作。因为包容操作约占 85%的运行时间，而我们不排除有更有效的包容操作实现方法，为慎重起见，我们同时给出“包容过程调用”和“包容操作”两个数据，相当于给出了包容基本操作的下界和上界；

注 e：子句表中子句数是指不被当时子句表中已有子句包容的归结式子句数和初始子句集中子句数之和；

注 f：由于支持集归结对变元数 30、子句数 129 的实例无有效结果，我们又对它测试了变元数 12、子句数 60 的实例；

注 g：对可满足实例，用局部搜索方法寻找划分子句集的解释（对语义归结）或尽可能小的支持集（对支持集归结）时已经给出实例的解，因而不必再进行归结过程。但切记：这不是归结法的威力，而是局部搜索方法的威力。在用吴方法之前用局部搜索方法做预处理会有同样结果。因此，应注意的是局部搜索方法对归结法证明不可满足实例的作用。

我们比较了吴方法与强约束变元低序下的语义归结。尽管归结法的两大基本操作归结和包容的计数中不包括单元归结中的有关操作，而吴方法的基本操作计数中包含了所有操作；此外，局部搜索方法的应用使语义归结的效率提高了一倍；但无论从基本操作计数还是从时间上看，吴方法的优势是显而易见的，而且优势很大。具体数据见实验 4。

实验 4　吴方法与语义归结的比较
变元数=30，子句数=129，实例数=50（其中可满足实例数=25）

	可满足实例		不可满足实例	
	时间	基本操作	时间	基本操作
语义归结 (强约束变元低序, 随机生成解释)	20	归结操作=23602 包容过程调用=20103 包容操作=5266310	20	归结操作=23708 包容过程调用=20249 包容操作=4932749
吴方法	0	6701	0	8980
语义归结 (强约束变元低序, 局部搜索解释)	0	0	8	归结操作=10804 包容过程调用=9124 包容操作=1798941

注 a：本实验与实验 3 是同一组实例；
注 b：有关基本操作的说明参见实验 1 和实验 3 的注。

4.3. 吴方法与 DP 算法的比较

DP 算法是目前求解 SAT 问题的完备算法中效率最高的一个，因此我们有必要把吴方法与 DP 算法进行比较，以全面评价吴方法的效率。

DP 算法经过几年来的集中研究，算法实现得比较好，其中分支变元的选取是关键。我们采用约束强的变元优先分支，变元约束强弱用变元以正文字出现的子句数和以负文字出现的子句数之乘积来估计，乘积越大，约束越强。吴方法的分支策略就是借鉴了 DP 算法的这一分支策略并利用特征列的信息而设计的。与吴方法有所不同的是，DP 算法在搜索树的各个节点上将动态调整变元序，而不是在搜索前一次定序后不再变化。实验 5 给出了 DP 算法和吴方法对变元数为 100、子句数为 430 的 100 个随机实例的实验数据。

实验 5　吴方法与 DP 算法比较 变元数=100，子句数=430，实例数=100（其中可满足实例数=54）						
	可满足实例		不可满足实例		全体实例	
	时间	节点数	时间	节点数	时间	节点数
DP 算法	0	155	0	471	0	301
吴方法	38	165	83	313	59	233

注：本组实例与实验 2 中同一规模的那组实例是相同的。

从实验 5 的数据看，吴方法的搜索树要比 DP 算法的搜索树小，但总的效率有很大差距。

我们进一步实验了把吴方法的特征列计算作为节点操作嵌入 DP 算法的想法，即在 DP 算法搜索树的每个节点上做完单元归结后，若既没发现矛盾也没找到解，则计算当前节点子问题(仍是子句集)的特征列，其中变元序在节点处按当前子问题强约束变元低序的策略动态确定。如果特征列计算发现矛盾或者找到解，则该节点计算可以比 DP 算法提前结束。如果如果特征列计算既没发现矛盾也没找到解，则继续按 DP 算法求解。实验 6 对变元数为 100、子句数为 430 的 100 个随机实例实验了这个想法。

实验 6　把吴方法的特征列计算嵌入 DP 算法，并与吴方法和 DP 算法比较 变元数=100，子句数=430，实例数=100（其中可满足实例数=54）						
	可满足实例		不可满足实例		全体实例	
	时间	节点数	时间	节点数	时间	节点数
DP 算法	0	155	0	471	0	301
吴方法	38	165	83	313	59	233
把吴方法的特征列计算嵌入 DP 算法	6	50	19	124	12	84

注：本实验与实验 5 是同一组实例。

从实验 6 的数据看，把吴方法的特征列计算嵌入 DP 算法后，节点数和时间均比吴方法要好；但与 DP 算法相比，尽管节点数下降很大，总的效率仍有差距。

提高吴方法的实现效率，关键在于提高特征列的计算效率。目前看来特征列的计算还有可能进一步减少无用操作，主要是改变特征列计算时

“牵一发而动全身”的特点，即在基列有所变化时，并不一定要重新计算所有多项式对此基列的余式，当然这需要进行必要的标记以及进一步的实验。

§5. 总结

本文在可读性变换思想的指导下，研究了吴方法求解 SAT 问题的特点，证明了在本文给出的输入变换下，吴方法求解 SAT 问题是一种以特征列计算为核心的有限制的子句归结过程，使吴方法作为求解 SAT 问题的一种全新的方法具有了可读性，这对于深入认识和有效实现吴方法对 SAT 问题的求解具有重要意义。

本文利用均匀随机 3-SAT 实例模型，通过全面详细的计算实验，研究了吴方法求解 SAT 问题的最优实现，并与归结法和 DP 算法进行了比较，对目前吴方法求解 SAT 问题的效率做了比较全面的评价。吴方法求解 SAT 问题与 DP 算法相比，效率上有一定差距；但与归结法相比，效率上有显著优势。因此，把吴方法应用于一阶谓词逻辑自动定理证明是一个值得研究的课题。

吴方法对于 SAT 问题求解是一条全新的路，提供了实质上不同的方法，而且今后吴方法的任何进展都可能对 SAT 问题求解产生进一步的有益影响和启发。同时，求解类似 SAT 这类已有一些解法的问题对于拓广吴方法的研究和应用领域也有积极作用。

本文对吴方法求解 SAT 问题的研究同时也是为了说明本文提出的可读性变换的算法设计思想。本文作者认为，虽然变换求解可以开阔解题思路，但是只有可读性变换才能真正提高问题求解的效率。可读性变换表示简洁，可以充分利用原问题的特性，可以有机结合、合理裁剪各种问题求解策略形成更有效的方法，还可以帮助我们认识不同问题以及不同求解方法之间的内在联系，这些优点本文通过吴方法求解 SAT 问题的整个过程给予了充分的说明。从实验结果看，吴方法求解 SAT 问题的效率还不太高，但是，这不是可读性变换的问题：如果不是采用可读性变换，

吴方法求解 SAT 问题就不只是效率高低的问题，而是可行不可行的问题了！

可读性变换是否总存在，这个问题不太好回答，因为很难区分不存在与存在而没发现。我们认为，可读性变换总是存在的，只是程度不同而已。但有一点可以肯定：可读性变换即使存在，也并不显然，这就需要我们首先树立可读性变换的思想，努力去挖掘、去发现可读性。本文通过吴方法求解 SAT 问题，示范了实现可读性变换的一种基本方法：通过设计合适的输入变换建立两种问题形式下基本操作的对应，然后利用问题特性对方法加以改造以进一步提高效率。目前，在各种组合优化问题形式下已经发展了大量求解方法，但还没有广泛地应用可读性变换的思想去认识、去发展、去统一这些方法。我们相信，可读性变换的算法设计思想可以大有作为！

参考文献

[Aarts et al., 1994] E. H. L. Aarts, P. J. M. van Laarhoven, J. K. Lenstra, and N. L. J. Ulder. A computational study of local search algorithms for job shop scheduling. ORSA Journal on Computing, 1994, 6(2): 118~125.

[Abramson & Yung, 1989] B. Abramson and M. Yung. Divide and conquer under global constraints: a solution to the N-queens problem. Journal of Parallel and Distributed Computing, 1989, 6: 649~662.

[Adorf & Johnston, 1990] H.-M. Adorf and M. D. Johnston. A discrete stochastic neural network algorithm for constraint satisfaction problems. In: Proceedings of the International Joint Conference on Neural Networks, San Diego, CA, 1990.

[白硕, 1992] 白硕. 有关“知道”的逻辑问题的代数方程表达初探. 第二届中国人工智能联合学术会议论文集, 杭州, 1992. 267~272.

[Bai & Liu, 1994] Shuo Bai and Tao Liu. An empirical study of the location of “really” hard 3-SAT instances. In: J. Crawford and B. Selman, eds. AAAI-94 Workshop on Experimental Evaluation of Reasoning and Search Methods, 1994. 1~3.

[Bai & Bu, 1996] Shuo Bai and Dongbo Bu. Modeling phase transition of random 3SAT problem. In: Selected Talks of International SAT Competition & Symposium’96, Beijing, 1996.

[Bentley, 1990] Jon Louis Bentley. Experiments on geometric traveling salesman heuristics. Computer Science Technical Report No. 151, AT&T Bell Laboratories, 1990.

[Bernhardsson, 1991] B. Bernhardsson. Explicit solutions to the N-queens problem for all N. SIGART Bulletin, 1991, 2(2): 7.

[Bickel, 1991] P. J. Bickel. 数理统计，李泽慧等译. 兰州：兰州大学出版社, 1991.

[卜东波 & 白硕, 1996] 卜东波, 白硕. 求解SAT问题的统一算法模型及子空间旋转策略在局部搜索算法中的应用. 国家智能计算机研究开发中心技术报告, 1996.

[Brady, 1977] M. Brady. *The Theory of Computer Science: A Programming Approach*. Chapman and Hall, 1977.

[Cha & Iwama, 1995] Byungki Cha and Kazuo Iwama. Performance test of local search algorithms using new types of random CNF formulas. In: Chris S. Mellish, eds. *Proceedings of the 14th International Joint Conference on Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann, 1995. 304~310.

[陈宝林, 1989] 陈宝林. 最优化理论和算法. 北京：清华大学出版社, 1989.

[常新功, 1994] 常新功. 基于统计的汉语文本识别知识后处理研究与实现：[硕士学位论文]. 北京：清华大学计算机系, 1994.

[Cheeseman et al., 1991] P. Cheeseman, B. Kanefsky, and W. M. Taylor. Where the really hard problems are. In: Proceedings of the 12th International Joint Conference on Artificial Intelligence. 1991. 331~337.

[陈希孺 & 柴根象, 1993] 陈希孺, 柴根象. 非参数统计教程. 上海：华东师大出版社, 1993.

[Cook, 1982] S. Cook. An overview of computational complexity. In: ACM Turing Award Lectures. Reading, MA: Addison-Wesley, 1987. 411~432.

[Dechter & Rish, 1994] R. Dechter and I. Rish. Directional resolution: the Davis-Putnam procedure, revisited. In: Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning. 1994. 134~145.

[Davis & Putnam, 1960] M. Davis and H. Putnam. A computing procedure for quantification theory. Journal of the ACM, 1960, 7: 201~215.

[Davis et al., 1962] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. Communications of the ACM, 1962, 5: 394~397.

[方开泰, 1994] 方开泰. 均匀设计与均匀设计表. 北京：科学出版社, 1994.

[方开泰 & 王元, 1996] 方开泰, 王元. 数论方法在统计中的应用. 北京：科学出版社, 1996.

[冯玉才 et al., 1995] 冯玉才, 黄文奇, 周旋. 求解雷达群监视目标群问题的拟物算法. 中国科学(A辑), 1995, 25(9): 982~988.

[Ferreira & Zerovnik, 1993] A. G. Ferreira and J. Zerovnik. Bounding the probability of success of stochastic methods for global optimization. Computers Math. Applic., 1993, 25(10/11): 1~8.

[Freuder, 1978] E. C. Freuder. Synthesizing constraint expressions. Communications of the ACM, 1978, 21(11): 958~966.

[Goldberg, 1989] David E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley, 1989.

[Golden & Stewart, 1985] B. L. Golden and W. R. Stewart. Empirical analysis of heuristics. In: E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, eds. The Traveling Salesman Problem. Chichester: John Wiley & Sons, 1985.

[Gu, 1992] J. Gu. Efficient local search for very large-scale satisfiability problems. *SIGART Bulletin*, 1992, 3(1): 8~12.

[Gu, 1993] J. Gu. Local search for satisfiability (SAT) problem. *IEEE Transactions on Systems, Man, and Cybernetics*, 1993, 23(4): 1108~1129.

[Gu, 1994] J. Gu. Global optimization for satisfiability (SAT) problem. *IEEE Transaction on Knowledge and Data Engineering*, 1994, 6(3): 361~381.

[Haken, 1985] A. Haken. The intractability of resolution. Theoretical Computer Science, 1985, 39: 297~308.

[华罗庚, 1984] 华罗庚. 华罗庚科普著作选. 上海：上海教育出版社, 1984.

[黄文奇 & 詹叔浩, 1979] 黄文奇, 詹叔浩. 求解Packing问题的拟物方法. 应用数学学报, 1979, (2): 176~180.

[黄文奇, 1989] 黄文奇. 求解Covering问题的拟物方法—NP难度问题的一个处理途径. 计算机学报, 1989, 12(8): 610~616.

[黄文奇 & 陈亮, 1991] 黄文奇, 陈亮. 求解有关空间利用的调度问题的拟物方法. 中国科学(A辑), 1991, (3): 325~331.

[黄文奇 et al., 1993] 黄文奇, 朱虹, 许向阳, 宋益民. 求解方格packing问题的启发式算法. 计算机学报, 1993, 16(11): 829~836.

[黄文奇 & 金人超] 黄文奇, 金人超. 求解SAT问题的拟物拟人算法—Solar. 1996.

[Johnson et al., 1988] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search. Journal of Computer and System Sciences, 1988, 37: 79~100.

[Johnson et al., 1989] David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, and Catherine Schevon. Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning. Operations Research, 1989, 37(6): 865~892.

[Johnson, 1990] David S. Johnson. Local optimization and the traveling salesman problem. In: Proceedings of the 17th Colloquium on Automata, Languages and Programming, Springer-Verlag, Berlin, 1990. 446~461.

[Johnson et al., 1991] David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, and Catherine Schevon. Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning. Operations Research, 1991, 39(3): 378~406.

[康立山 et al., 1994] 康立山, 谢云, 尤矢勇, 罗祖华. 非数值并行算法(第一册)：模拟退火算法. 北京：科学出版社, 1994.

[Kapur & Lakshman, 1992] D. Kapur and Y. N. Lakshman. Elimination theory: an introduction, Chapter 2 in: B.R.Donald, D.Kapur and J.L.Mundy, eds. Symbolic and Numerical Computation for Artificial Intelligence. Academic Press, 1992.

[Kapur & Narendran, 1985] D. Kapur and P. Narendran. An equational approach to theorem proving in first-order predicate calculus. In: Proceedings of the 10th International Joint Conference on Artificial Intelligence, 1985. 1146~1153.

[Kask & Dechter, 1995] K. Kask and R. Dechter. GSAT and local consistency. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1995. 616~622.

[Kernighan & Lin, 1970] B. W. Kernighan, S. Lin. An efficient heuristic procedure for partitioning graphs. The Bell System Technical Journal, 1970, 49: 291~307.

[Kirpatrick et al., 1983] S. Kirpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by simulated annealing. Science, 1983, 220(4598): 671~680.

[Kolen & Pesch, 1994] A. Kolen and E. Pesch. Genetic local search in combinatorial optimization. Discrete Applied Mathematics, 1994, 48: 273~284.

[李久坤, 1996] 李久坤. “平均冒尖性”不能作为比较正交设计和均匀设计的准则. 数理统计与管理, 1996, 15(5): 48~53.

[李未 & 黄文奇, 1994] 李未, 黄文奇. 一种求解合取范式可满足性问题的数学物理方法. 中国科学(A辑), 1994, 24(11): 1208~1217.

[梁东敏 et al., 1996] 梁东敏, 吴晔, 马绍汉. 1996北京SAT问题算法比赛程序.

[林成江, 1995] 林成江. 并行处理系统仿真与负载平衡算法的研究：[博士学位论文]. 北京：清华大学计算机系, 1995.

[Lin & Kernighan, 1973] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. Operations Research, 1973, 21: 498~516.

[刘涛, 1995] 刘涛. 约束满足问题: 算法和复杂性：[博士学位论文]. 北京：中科院计算所,1995.

[刘婉如 et al., 1995] 刘婉如, 庞善起, 张里千, 张建方. 关于正交设计与均匀设计的比较(II). 数理统计与管理, 1995, 14(2): 41~49.

[刘勇 et al., 1995] 刘勇, 康立山, 陈毓屏. 非数值并行算法(第二册)：遗传算法. 北京：科学出版社, 1995.

[柳常青, 1997] 柳常青. 基于不完全算法复杂性的一般优化算法研究：[博士学位论文]. 北京：清华大学计算机系, 1997.

[Lourenco, 1995] Helena Ramalhinho Lourenco. Job-shop scheduling: computational study of local search and large-step optimization methods. European Journal of Operational Research, 1995, 83: 347~364.

[Lu, 1996] Lu Weifeng. Experimental study on strategies of combining SAT algorithms. In: Selected Talks of International SAT Competition & Symposium'96, Beijing, 1996.

[Martin et al., 1992] O. Martin, S. W. Otto, and E. W. Felten. Large-step Markov chains for the TSP incorporating local search heuristics. Operations Research Letters, 1992, 11: 219~224.

[Minton et al., 1990] S. Minton, M. D. Johnston, A. B. Philips, and P. Laird. Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method. In: Proceedings of AAAI-90, 1990, 17~24.

[Mitchell et al., 1992] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distribution of SAT problems. In: Proceedings of AAAI-92, 1992, 459~465.

[Mitchell & Levesque, 1996] D. G. Mitchell and H. J. Levesque. Some pitfalls for experimenters with random SAT. Artificial Intelligence, 1996, 81: 111~126.

[Morris, 1993] P. Morris. The breakout method for escaping from local minima. In: Proceedings of AAAI-93, 1993, 40~45.

[Morris & Wong, 1991] Morris, R. J. T. and Wong, W. S. Systematic choice of initial points in local search: extensions and application to neural networks. Information Processing Letters, 1991, 39: 213~217.

[Papadimitriou & Steiglitz, 1982] C. H. Papadimitriou, K. Steiglitz. Combinatorial Optimization. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[Robinson, 1965] J. A. Robinson. A machine-oriented logic based on the resolution principle. Journal of the ACM, 1965, 12(1): 23~41.

[Selman et al., 1992] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In: Proceedings of AAAI-92, 1992, 440~446.

[Selman & Kautz, 1993] B. Selman and H. Kautz. Domain-independent extensions to GSAT: solving large structured satisfiability problems. In: Proceedings of AAAI-93, 1993, 290~295.

[Selman et al., 1994] B. Selman, H. A. Kautz, and B. Cohen. Noise strategies for improving local search. In: Proceedings of AAAI-94, 1994. 337~343.

[石赫, 1994] 石赫. 吴文俊消元法. 机械化数学讲座之一, 中科院系统所, 1994.5.

[斯华龄, 1993] 斯华龄. 电脑人脑化：神经网络—第六代计算机(普及本). 北京：北京大学出版社, 1993.

[Slagle, 1967] J. Slagle. Automatic theorem proving with renamable and semantic resolution. Journal of the ACM, 1967, 14: 687~697.

[Sobol, 1979] I. M. Sobol. On the systematic search in a hypercube. SIAM Journal on Numerical Analysis, 1979, 16(5): 790~793.

[Sosic & Gu, 1990] R. Sosic and J. Gu. A polynomial time algorithm for the N-queens problem. SIGART Bulletin, 1990, 1(3): 7~11.

[Sosic & Gu, 1991] R. Sosic and J. Gu. 3,000,000 queens in less than one minute. SIGART Bulletin, 1991, 2(2): 22~24.

[Sun et al., 1996] Sun Jigui, Liu Ruisheng, and Chen Rong. Using Wu's Method to solve satisfiability problems. In: Selected Talks of International SAT Competition & Symposium'96, Beijing, 1996.

[滕弘飞 et al., 1993] 滕弘飞, 刘义军, 葛文海, 孙大新, 钟万勰. 旋转锥体空间中圆柱体群的布局优化. 计算机学报, 1993, 16(7): 519~525.

[滕弘飞 et al., 1994] 滕弘飞, 孙守林, 葛文海, 钟万勰. 转动圆桌平衡摆盘—带平衡性能约束的Packing问题. 中国科学(A辑), 1994, 24(7): 754~760.

[Traub & Wozniakowski, 1994] J. F. Traub and H. Wozniakowski. Breaking intractability. Scientific American, January 1994, 102~107.

[Ulder et al., 1990] N. L. J. Ulder, E. H. L. Aarts, Hans-Jurgen Bandelt, P. J. M. van Laarhoven, and Erwin Pesch. Genetic local search algorithms for the traveling salesman problem. In: Lecture Notes in Computer Science 496. Berlin: Springer-Verlag , 1991. 409~416.

[Whitley et al., 1996] D. Whitley, S. Rana, J. Dzubera, and K. E. Mathias. Evaluating evolutionary algorithms. Artificial Intelligence, 1996, 85: 245~276.

[Wong & Morris, 1989] Wong, W. S. and Morris, R. J. T. A new approach to choosing initial points in local search. Information Processing Letters, 1989, 30: 67~72.

[Wos et al., 1965] L. Wos, D. Carson, and G. Robinson. Efficiency and completeness of the set-of-support strategy in theorem proving. Journal of the ACM, 1965, 12: 536~541.

[Wos, 1988] L. Wos. Automated reasoning: 33 basic research problems. Prentice Hall, 1988.

[吴文俊, 1984] 吴文俊. 几何定理机器证明的基本原理(初等几何部分). 北京：科学出版社, 1984.

[项可风 & 吴启光, 1989] 项可风, 吴启光. 试验设计和数据分析. 上海：上海科学技术出版社, 1989.

[Yannakakis, 1990] M. Yannakakis. The analysis of local search problems and their heuristics. In: Proceedings of the 7th Annual Symposium on Theoretical Aspects of Computer Science, 1990. 298~311.

[Yugami et al., 1994] N. Yugami, Y. Ohta, and H. Hara. Improving repair-based constraint satisfaction methods by value propagation. In: Proceedings of AAAI-94, 1994, 344~349.

[张里千, 1995] 张里千. 关于正交设计与均匀设计的比较(I). 数理统计与管理, 1995, 14(1): 25~30.

[周芝英 et al., 1995] 周芝英, 刘婉如, 张里千. 关于正交设计与均匀设计的比较(III). 数理统计与管理, 1995, 14(4): 37~42.

作者完成的论文

贺思敏, 卢旭光, 张钹. 局部搜索多初始点选择的新策略——划分策略的性能分析. 计算机学报, 1997. (已接受)

贺思敏, 张钹. 用吴方法求解可满足性问题. 计算机学报, 1997. (已接受)

Simin He and Bo Zhang. Solving SAT by Readable Transform of Wu's Method. accepted by *Poster Session of The 15th International Joint Conference on Artificial Intelligence*, 1997.