基于序列模型的优化
General Algorithm Configuration
(extended version)

Frank Hutter, Holger H. Hoos and Kevin Leyton-Brown

不列颠哥伦比亚省大学，2366大购物中心，温哥华BC，V6T 1Z4，加拿大
{hutter,hoos,kevinlb}@cs.ubc.ca

抽象的。用于硬计算问题的最先进的算法通常可以修改许多参数，以改进以提高实证性能。然而，手动探索所得到的参数设定的组合空间是乏味的并且倾向于导致不令人满意的结果。最近，解决该算法配置问题的自动化方法导致了用于解决各种问题的领域的大量改进。一个有希望的方法构建了显式回归模型，以描述目标算法性能对参数设置的依赖性;然而，到目前为止，这种方法仅限于单个实例上的几个数值算法参数的优化。在本文中，我们首次扩展了此范例的算法配置问题，允许许多分类参数和实例的优化。我们通过优化本地搜索和树搜索解决方案来实验验证我们的新算法配置步骤，用于命题可满足问题（SAT），以及商业混合整数编程（MIP）求解器CPLEL。在这些实验中，我们的程序产生了最先进的性能，并且在许多情况下优于先前的最佳配置方法。

1 Introduction

硬计算问题的算法 – 无论是基于本地搜索还是树搜索 –
通常都是高度参数化的。本地搜索中的典型参数包括邻居，禁忌任期，随机步道的百分比，以及迭代本地搜索中的扰动和验收标准。树搜索中的典型参数包括关于预处理，分支规则的决策，在每个搜索节点（例如，计算剪切或下界），学习何时执行重新启动时，可以执行多少。作为一个突出的例子，商业混合整数编程求解器IBM ILOG
CPLEX具有76个参数，与其搜索策略有关[1]。优化这些参数的设置可以大大提高性能，但手动这样做是乏味的，通常是不切实际的。

解决此算法配置问题的自动程序是有用的
在各种背景下。他们最突出的用例是从某些应用程序（“离线”的培训实例集上优化参数（作为算法开发的一部分），以便在练习中使用算法（“在线”）时提高性能。

因此，算法配置为机器时间交易人类时间，并自动化将手动执行的任务。算法的最终用户还可以应用算法配置过程（例如，内置于CPLEX版本11及更高版本中的自动调谐工具），以在兴趣的问题实例上配置现有的高性能算法。

算法配置问题可以正式说明如下：给出了一个

参数化算法A（目标算法），问题实例I和成本度量C的集合（或分布），查找最小化C上的参数设置。成本度量C通常基于解决a所需的运行时问题实例，或者在优化问题的情况下，在给定时间预算中实现的解决方案质量。已经提出了各种自动化程序来解决该算法配置问题。现有方法是否与显式模型用于描述目标算法性能对参数设置的依赖性不同。

无模型算法配置方法相对简单，可以应用

开箱即用，最近导致了各种约束编程域的实质性改进。这项研究返回了20世纪90年代初[2,3]，最近一直在获得势头。一些方法侧重于优化数值（即，整数或实值）参数（参见，例如[4,5,6]），而其他方法还针对分类（即，离散值和无序）域[7,
8,9,10,11]。最突出的配置方法是赛车算法F−Race
[7]和我们自己的迭代本地搜索算法参数[8,9]。最近的竞争对手是遗传算法GGA [10]。
F−Race及其扩展已被用于优化各种高性能算法，包括迭代本地搜索和蚁群优化程序，用于时间表任务和旅行销售人员问题[12,6]。我们自己的小组使用了参数来配置高度参数化的树搜索[13]和本地搜索求解器[14]，用于命题可靠性问题（SAT），以及用于混合整数编程（MIP）的若干求解器，基本上推动了各种类型的艺术。值得注意的是，通过优化CPLEX的76个参数 − 最突出的MIP求解器 −
我们通过默认值和CPLEX调整工具返回的配置实现了高达50倍的加速度[1]。

虽然上述实际应用的进展是基于的

无模型优化方法，基于模型的方法的最新进展承诺导致下一代算法配置过程。基于模型的优化（SMBO）在拟合模型之间迭代并使用它们进行选择，以便选择要调查的配置。它提供了在观察到的参数设置和外推到以前看不见的参数空间区域之间的内插性能的吸引人前景。它还可以用于量化每个参数和参数交互的重要性。然而，从统计信息中的“黑匣子功能优化”文献中的基础（参见，例如，[15]），SMBO继承了一系列限制，不适合自动算法配置设置。这些限制包括对确定性目标算法的关注;使用昂贵的初始实验设计;依赖计算昂贵的模型;并假设所有目标算法运行的都具有相同的执行成本。尽管最近的近期进步[16,17,18,19]，所有关于SMBO的公布工作仍然有三个关键限制，防止其用于一般算法配置任务：（1）它只支持数值

参数;（2）仅优化单个实例的目标算法性能;（3）它缺乏用于终止表现不良的目标算法早期运行的机制。

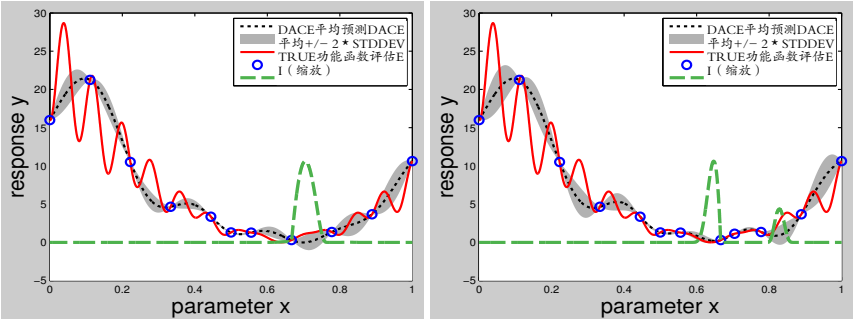
本文的主要贡献是删除这些SMBO限制的前两个 -
因此，使SMBO适用于许多分类参数和基准实例集的通用算法配置问题。具体而言，我们基因对SMBO框架的四个组成部分进行了ralize和基于它们 -
定义了一般算法配置的两个新型SMBO实例化：简单的无模型随机在线自适应赛车（咆哮）程序和更复杂的序列模型基于算法配置（SMAC）方法。这些方法还尚未实施用于执行目标算法不良的早期终止标准（例如，例如，Paramills的自适应封盖机制[9]）;因此，到目前为止，我们希望他们在具有大笔累积时的某些配置方案上表现不佳。在彻底的实验分析中，对于具有小批量的广泛的17场景（涉及本地搜索和树搜索的优化SAT求解器，以及商业MIP Solver
CPLEX），SMAC确实比较了两种最突出的方法算法配置：参数[8,9]和GGA [10]。

本文的其余部分的结构如下。第2节描述了SMBO
框架和以前的SMBO工作。第3节和第4节概括了SMBO的组件，以解决一般算法配置方案，分别定义咆哮和SMAC。第5节通过算法配置实验将咆哮和SMAC与现有技术的现有状态进行比较。第6节结束了论文。

2现有基于模型的优化的工作（SMBO）

基于模型的优化方法构造一种预测性能的回归模型（通常称为响应面模型），然后使用该模型进行优化。基于模型的优化（SMBO）在拟合模型和基于此模型的附加数据之间迭代。在参数优化的上下文中，该模型适用于训练集{（θ1，o1），。。。，（θn，开启）}其中参数配置θi=（θi，1，....。，θi，d）是目标算法的d参数的完整实例，并且Oi是在使用配置θi运行时的目标算法的观察性能。给定新配置θn+
1，模型旨在预测其在+ 1上的性能。

基于模型的优化（SMBO）迭代建筑模型与
收集其他数据。我们在图1中示出了一个简单的SMBO程序。考虑具有单个连续参数X的确定性算法A，并让A的运行时间作为其参数的函数由图1（a）中的实线描述。
SMBO搜索最小化此运行时的X的值。这里，通过运行由图1（a）中的圆圈指示的参数值来初始化它。接下来，SMBO适合收集的数据响应表面模型;高斯过程（GP）模型[20]是最常见的选择。图1中的黑色虚线表示在圆圈给出的数据上培训的GP模型的预测平均值，而周围的阴影区域量化了预测中的不确定性;这种不确定性随距离训练数据的距离而增长。
SMBO使用这种预测性能模型来选择A的下一步运行的有希望的参数配置。预先预测配置并和/或位于模型仍然不确定的区域中。这两个目标是



(a) SMBO, step 1 (b) SMBO, step 2

图1.
SMBO的两个步骤用于优化1D功能。真实功能显示为实线，圆圈表示我们的观察。虚线表示无噪声高斯工艺模型（“DACE”模型）的平均预测，灰色区域表示其不确定性。预期改进（可视化缩放）被显示为虚线。

结合在所谓的预期改进（EI）标准中，在低预测平均值和高预测方差的区域中很高（参见图1（a）中的光线虚线;在等式3中给出了EI的精确公式第4.3节）。SMBO选择具有最大EI的配置（此处，x = 0.705），使用它使用它，并根据结果更新其模型。在图1（b）中，我们展示了这个新的数据点如何改变模型：注意x = 0.705的附加数据点，它周围的大大降低的不确定性，并且大型ei的区域现在被分成两个。

虽然我们的示例捕获了SMBO的本质，但最近的实际SMBO Instantia− Tions包括更复杂的机制，用于处理算法性能中的随机性，并降低计算开销。算法框架1提供了我们在本文中采用的时界SMBO框架的一般结构。它首先使用一些初始参数配置运行目标算法，然后迭代三个步骤：（1）使用现有数据拟合响应面模型;（2）选择有前途的配置列表;（3）在达到给定时间绑定之前运行目标算法（某些）所选配置。由于拟合模型并选择有希望的配置，这次绑定与组合的开销TModel + Tei有关。

SMBO已经是全球实验设计统计文献的根源连续（“黑匣子”）功能优化。最值得注意的是Jones等人的有效的全局优化（EGO）算法。[15];这基本上是我们上面简单示例中使用的算法。自我限于优化无噪声功能的连续参数（即确定性算法的性能）。统计社区的后续工作包括一种方法来优化跨多个环境条件[21]的功能，以及用于处理噪声功能的顺序克里格优化（SKO）算法（即，在我们的上下文，随机算法中），由huang等。[22]。与后者工作并行，Bartz−Beielstein等。[16,17]是第一个使用自我方法来优化算法性能的方法。他们的顺序

算法框架1：基于顺序模型的优化（SMBO）R跟踪到目前为止所执行的所有目标算法及其性能（即，SMBO的训练数据{（[θ1，x1]，o1），...，（[θn，xn]，上）}），m是smbo的模型，　θew是有希望配置的列表，而TFIT和TSELECT分别是拟合模型所需的运行时间，并分别选择配置。

输入：目标算法A具有参数配置空间θ;实例设置π;成本指标C.

Output: Optimized (incumbent) parameter configuration, $\boldsymbol{\theta}_{inc}$

1 $[\mathbf{R}, \boldsymbol{\theta}_{inc}] \leftarrow$ *Initialize*($\boldsymbol{\Theta}$, Π);
2 **repeat**
3 　$[\mathcal{M}, t_{fit}] \leftarrow$ *FitModel*($\mathbf{R}$);
4 　$[\vec{\boldsymbol{\Theta}}_{new}, t_{select}] \leftarrow$ *SelectConfigurations*($\mathcal{M}$, $\boldsymbol{\theta}_{inc}$, $\boldsymbol{\Theta}$);
5 　$[\mathbf{R}, \boldsymbol{\theta}_{inc}]$　　*Intensify*($\vec{\boldsymbol{\Theta}}_{new}$, $\boldsymbol{\theta}_{inc}$,　　, $\mathbf{R}$, $t_{fit} + t_{select}$, Π, $\hat{c}$);
6直到配置耗尽的总时间预算;
7 **return** $\boldsymbol{\theta}_{inc}$;

参数优化（SPO）工具箱 – 在进化算法中接受了相当大的关注 –
提供了许多有助于手动分析和优化算法参数的功能;它还包括自动SMBO过程，用于在单个实例上优化连续参数。通过比较Sko VS
Spo，我们在SMBO开始了自己的工作，研究了四个SMBO组件的选择[18]。我们展示了组件加强了最重要的问题，并在我们的spo
+算法中改进了它[18]。随后，我们展示了如何通过近似GP模型来减少施工和使用响应面模型所产生的开销。我们还通过在整个优化过程中交错来删除了昂贵的初始设计的需要，而是利用不同算法运行的不同时间采取不同的时间。由此产生的时隙SPO变量TB−SPO是给定参数优化的第一个SMBO方法，给定用户指定的时间预算[19]。虽然它在某些域上显示出明显超越的发行者，但它仍然限于单个问题实例的连续算法参数的优化。在下文中，我们概括了时间限定的SMBO框架（其中TB−SPO是即时）的组件，扩展其范围以解决许多分类参数和基准实例集的常规算法配置问题。

3随机在线侵略性赛车（咆哮）

在本节中，我们首先概括了SMBO的加强程序来处理多个实例，然后引入咆哮，这是一种基于这种新的强化机制的非常简单的无模型算法配置过程。

步骤2：强化（ θ Ew， θ inc，m，r，tintenify， π ，c）c（ θ ， π '）表示基于r中的运行在 π ' π π sub π 上的 θ 上的经验成本;MAXR是一个参数，在我们的所有实验中设置为2 000

输入：参数设置的序列，以评估， θ new;现任参数设置， θ inc;模型，m;目标算法的序列运行，R;束缚，传输;实例设置， π ;成本度量，c

输出：更新的目标算法序列运行，R;现任参数设置， θ inc

1 **for** $i := 1, \ldots,$ *length*$(\boldsymbol{\Theta}_{new})$ **do**
2 $\quad \boldsymbol{\theta}_{new} \leftarrow \vec{\boldsymbol{\Theta}}_{new}[i]$;
3 $\quad$ **if** $\mathbf{R}$ *contains less than* $maxR$ *runs with configuration* $\boldsymbol{\theta}_{inc}$ **then**
4 $\quad\quad$ equal number of runs using $\boldsymbol{\theta}_{inc}$ and π'
$\quad\quad$ 而不是使用 θ inc和任何其他 π ' ∈ π };
5 π ←←在 π '随机均匀地采样;
6 S←种子，随机均匀拉伸;
$\quad$ └
8 $\quad N \leftarrow 1$;
9 $\quad$ **while** *true* **do**
10 $\quad\quad$ $S_{missing}$ instance, seed pairs for which $\boldsymbol{\theta}_{inc}$ was run before, but not $\boldsymbol{\theta}_{new}$;
11 Storun←随机散发的大小分钟（N，|散发）;
13 $\quad\quad$ $S_{missing}$ $S_{missing}$ S_{torun};
14 π Common←实例我们之前运行 θ inc和 θ new;
16 $\quad\quad$ **else if** $S_{missing} = \emptyset$ **then** $\boldsymbol{\theta}_{inc} \leftarrow \boldsymbol{\theta}_{new}$; **break**;
17 $\quad\quad$ **else** N 2 N;
18如果在此过程中花费的时间超过了Tintenify，那么I≥2则断裂;
19 $[\mathbf{R}, \boldsymbol{\theta}_{inc}]$

3.1概括I：多个实例的强化机制

任何算法配置过程的一个关键组件是所谓的替换机制，它控制了每个配置执行多少评估，以及何时信任足够的配置，以使其成为新的最新的已知配置（现任）。配置实例集的算法时，我们还需要确定每个运行中使用哪个实例。为了解决这个问题，我们概括了TB-SPO的强化机制。我们的新程序实现了一个方差减少机制，反映了洞察力，当我们在多个实例上比较两个参数配置的经验成本统计数据时，如果我们使用相同的n个实例来计算两个估计，则此比较中的方差较低。

步骤2更精确地定义了这种新的强化机制。它需要
输入有希望配置的列表， θ new，并将其与当前的现任配置进行比较，直到达到此比较阶段的时间预算

¯1如果在 θ new的第一个配置之后已经达到了该预算，则更新
用来;有关解释，请参阅第4.3节的最后一段。为什么。

在新配置的每个比较中，θ new，对现有的，θ inc，我们首先使用随机选择的 静音组合来执行用于现任的额外运行。然后，我们迭代地执行与θ new（使用加倍方案）运行，直到θ new的经验性能比θ inc更差（在这种情况下，我们拒绝θ new），或者我们对θ inc的θ new的许多运行执行，它仍然至少是和θ inc一样好（在这种情况下，我们将现任者改为θ new）。θ new的 静脉组合从现任者已经运行的那些随机进行了均匀地进行采样。然而，过程2中的每一个比较都基于不同的随机选择的实例和种子子集，而Focusedils的程序“更好”使用固定的排序，它可以非常敏感。

3.2 Defining ROAR

我们现在定义随机的在线攻击赛车（咆哮），一个简单的无模型瞬间，即一般的SMBO框架（参见算法框架1）.2这项令人惊讶的有效方法在随机选择均匀的参数配置，并迭代地将它们与当前现任者进行比较使用我们的新强化机制。我们认为咆哮为赛车算法，因为它只在需要确定它是否具有竞争力时才运行每个候选配置。它获得了名称，因为当时选择了一组候选人，每位候选人都被接受或在线拒绝，并且在收集足够的数据之前，我们正在进行这一在线决策，以支持统计上的结论。更正式的是，作为SMBO框架的实例化，咆哮完全由四个组件，FitModel，SelectConfigurations和Chrocenify指定。初始化执行单个运行，使用目标算法的默认参数配置（或者如果没有默认可用），则在随机选择的实例上统一地选择。由于咆哮是无模型的，其FitModel程序只是返回一个恒定的模型，从未使用过。
SelectConfigurations从参数空间随机返回一个均匀的单个配置，并且加强如过程2中所述。

4个基于模型的算法配置（SMAC）

在本节中，我们介绍了一般的SMBO框架的第二个更复杂的实例化：基于序列模型的算法配置（SMAC）。SMAC可以被理解为咆哮的延伸，可以基于模型而不是随机均匀地选择配置。它实例化以与咆哮的方式相同的方式初始化和强化。在这里，我们讨论我们在SMAC中使用的新模型类以支持分类参数和多个实例（分别为4.1和4.2）;然后，我们描述SMAC如何使用其模型来选择有希望的参数配置（第4.3节）。最后，我们证明了咆哮和SMAC的收敛结果（第4.4节）。

¯2我们以前考虑了基于不太强大的强化的随机抽样方法

mechanisms; see, *e.g.*, RANDOM* defined in [19].

4.1泛化II：分类参数的模型

我们所知的所有现有SMBO方法中的模型都仅限于数值参数。在本节中，为了处理分类参数，我们适应最突出的先前使用的模型类（高斯随机过程模型）并向SMBO介绍了模型类随机林。

GP模型的加权汉明距离核函数。最新的基于模型的优化工作[15,16,18]使用高斯随机过程模型（GPS;见[20]）。GP模型依赖于参数化内核功能k：θ × θ →R
+，它指定两个参数配置之间的相似性。以前的数字参数的SMBO方法通常选择GP内核功能

$$k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \exp\left[\sum_{l=1}^{d}(-\lambda_l \cdot (\boldsymbol{\theta}_{i,l} - \boldsymbol{\theta}_{j,l})^2)\right], \tag{1}$$

其中λ1，。。。，λd是内核参数。

对于分类参数，我们定义了一个新的类似内核。而不是测量
（加权）平方距离，它计算（加权）汉明距离：

$$k_{cat}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \exp\left[\sum_{l=1}^{d}(-\lambda_l \cdot [1 - \delta(\theta_{i,l}, \theta_{j,l})])\right], \tag{2}$$

其中Δ是kronecker delta函数（如果它的两个参数是相同的，否则为零）。

对于连续参数PCONT和分类参数PCAT的组合，
我们应用组合的内核

$$K_{mixed}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \exp\left[\sum_{l\in\mathcal{P}_{cont}}(-\lambda_l \cdot (\theta_{i,l} - \theta_{j,l})^2) + \sum_{l\in\mathcal{P}_{cat}}(-\lambda_l \cdot [1 - \delta(\theta_{i,l}, \theta_{j,l})])\right].$$

虽然kmixed是等式1中标准高斯内核的直接概括，但我们不知道任何先前使用此内核或证明它确实是一个有效的内核功能.3我们在附录中提供此证明。由于高斯随机过程是基于内核的学习方法，并且由于kmixed是一个有效的内核函数，因此它可以在不改变GP模型的任何其他组件的情况下交换Gaussian内核。在这里，我们使用GP模型的相同投影过程（PP）近似[20]，如TB−SPO [19]中。

随机森林。我们在SMAC中使用的新默认模型基于随机林[24]，是回归和分类的标准机器学习工具。随机森林是回归树的集合，类似于决策树，但有真实

¯3 COUTO [23]给出了与它也是相关的分类数据的递归内核功能

基于汉明距离。

值（此处：目标算法性能值）而不是叶子的类标签。已知回归树对分类输入数据执行良好;实际上，它们已经被用于建模启发式算法的性能（在运行时和解决方案质量方面）（例如，[25,26]）。随机森林分享这种好处，通常会产生更准确的预测[24];他们还允许我们在给定的预测中量化我们的不确定性。我们将一个随机森林作为一组B回归树，每个都基于从整个训练数据集的重复随机采样的n个数据点{（θ1，o1）。。。，（θn，开）}。在每棵树的每个分割点处，D算法参数的 d·p 的随机子集被认为是有资格的;分流比P是一个参数，我们留下了默认值P = 5/6。另一个参数是nmin，如果进一步拆分，则需要在节点中所需的最小数据点;我们使用标准值nmin = 10.最后，我们将树的数量设置为b = 10以保持计算开销小.4我们计算随机森林的预测均值μ θ和方差σ2

θ为新配置θ作为经验

其个体树木对θ的预测的平均值和方差。通常，参数配置θn+1的树预测是在树下传播θn+1时的叶子中的数据点的均值。我们改进了这种机制，而是预测该数据的用户定义的成本度量，例如，该叶子中的数据点的中值。

成本度量的转变。通常可以通过转换成本度量来改善模型拟合。在本文中，我们专注于最小化算法运行时。以前的研究预测算法运行时已经发现对数变换大大提高了模型质量[27]，我们在本文中使用了对数转换的运行时数据;也就是说，对于运行时RI，我们使用OI = LN（RI）。
（SMAC也可以应用于优化其他成本指标，例如解决方案质量算法在固定的运行时获得算法;其他转换可能对其他指标进行更高效。但是，我们注意到在某些型号中，这种转换隐含地改变了成本公制用户旨在优化。例如，采用一个简单的情况，其中只有一个参数配置θ，我们测量了运行时间（R1，....。。，r10）=（21,22，...。。。。虽然这些运行的真正算术平均值大致205，但是使用日志转换在此数据上培训的GP模型将预测exp的平均值（（log（r1），...，log（r10））））≈45.这是因为日志的算术平均值是几何平均值的日志：

$$\text{geometric mean} = \sqrt[n]{\prod_{i=1}^{n} x_i} = \left[\exp\left(\sum_{i=1}^{n}\log(x_i)\right)\right]^{(1/n)}$$
$$= \exp\left[\frac{1}{n}\sum_{i=1}^{n}\log(x_i)\right] = \exp\left(\text{mean of logs}\right).$$

对于GPS来说，目前尚不清楚如何解决这个问题。通过“未转发”数据，计算用户定义的成本度量，避免通过计算树的叶片中的预测来避免在随机林中避免此问题。然后再次转换结果。

¯4这三个参数的优化可能进一步提高性能。我们计划

在SMAC应用于优化其自己的参数的情况下研究这一点。

4.2概括III：问题实例集模型

有几种可能的方法可以扩展SMBO的模型来处理多个实例。最简单的是，可以使用固定的N个实例用于目标算法运行的每次评估，报告聚合性能。但是，N：小N没有良好的固定选择导致较差的概括到测试数据，而大n导致每种评估的成本上的禁止的n倍降低。（这是Paramils Instantiation
Basicils（n）[8]所面临的相同问题，而是明确地将关于该实例的信息集成到我们的响应曲面模型中。给定具有描述每个训练问题实例πi∈π的特征xi的传染媒介，我们学习一个联合模型，用于预测参数配置和实例特征的组合的算法运行时。然后我们跨实例聚合这些预测。

实例功能。对经验硬度模型的现有工作已经证明，可以基于给定问题实例的特征来预测算法运行时。最值得注意的是，已经利用这种预测来构建基于投资组合的算法选择机制，例如Satzilla [29,27]。

对于CNF公式形式的SAT实例，我们使用了126个功能，包括
基于图形表示的功能，LP放松，DPLL探测，本地搜索探测，条款学习和调查传播。所有功能都列在图2中。对于MIP实例，我们计算了39个功能，包括基于图形表示，LP松弛，目标函数和线性约束矩阵的特征。所有功能都列出了图3中。为了降低学习的计算复杂性，我们应用了主成分分析（参见，例如[30]），将特征矩阵投影为由七个正交向量跨越的较低维子空间它具有最大的方差。

对于新域，没有定义功能，SMAC仍然可以
使用空特征集或简单的域 -
独立特征，例如实例大小或算法的默认设置的性能（基于初步实验，似乎是一个令人惊讶的有效特征）。请注意，与PER-Mation方法相比，诸如Satzilla
[29,27]，实例特征仅适用于训练实例：算法配置的最终结果是一个无需计算功能的单个参数配置测试实例。作为推论，特征计算所需的时间在算法配置中不是至关重要的，因为它在每个实例方法中：在每个实例方法中，必须将特征计算作为求解测试实例所需的时间的一部分，但是在算法配置中，根本没有计算测试实例的功能。实际上，培训实例的功能可能是对这些培训实例进行广泛的离线分析的结果，或者甚至可以从文献中获取。计算我们在此处使用的功能平均为SAT域花了30秒，并且MIP域为4.4秒。

预测跨实例的性能。到目前为止，我们已经讨论了参数配置θi的对（θi，oi）培训的模型及其观察到的性能OI。现在，我们将此数据扩展为包含实例功能。让xi表示矢量

Problem Size Features:

1.–2。orig–的变量和子句数量
inal formula: denoted v and c, respec-

3.–4。之后的变量和条款的数量
使用卫星简化：分别表示V'和C'

5.–6。减少变量和条款

c'

7.变量与条款的比率：v'/ c'

Variable-Clause Graph Features:

8.–12。变量节点度统计：均值，
变化系数，min，max和杂交

13.–17。子句节点学位统计：均值，
变化系数，min，max和杂交

Variable Graph Features:

18–21。节点度统计：均值，变异
系数，min和max

22.–26. mean, variation coefficient,
最小，最大和熵

27.–31. : mean, variation
系数，分钟，最大和熵

Clause Graph Features:

32–36。节点度统计：均值，变异
系数，分钟，最大和熵

Balance Features:

37.–41。积极对负面文字的比例
每个条款：平均值，变异系数，min，ma
x和熵

42.–46。阳性与消极的比率 –
每个变量的范围：平均值，变化系数，mi
n，max和熵

47.–49。一组机构，二元和三元
clauses

邻近喇叭公式：50.喇叭条件的
分数

51.–55。喇叭子句的出现次数
对于每个变量：平均值，变异系数，min
，max和熵

DPLL Probing Features:

56.–60。单位传播数：计算
at depths 1, 4, 16, 64 and 256

61.–62。搜索空间大小估计：平均深度
矛盾，估计节点数量的日志

LP-Based Features:

63.–66。整数松弛载体：平均值，变异共同

67.
LP解决方案68中整数变量的比率。LP解决方案的目标函数值

本地搜索探测功能，基于运行每个SAP和GSAT
的2秒：

69.–78。最佳本地分钟的步数
在赛跑中：平均值，中位数，变异系数，
10和90百分位数

79.–82。跑步中最好的平均改善：
每步改善变化的均值和系数，以最佳解决
方案

83.–86。由于第一个LO–由于提高的分数
CAL最小值：平均值和变化系数

87.–90。数量变异系数
每个局部迷你妈妈中的不满意的条款：平
均值和变化系数

子句学习功能（基于2秒的运行Zchaff
Rand）：

91.–99。学习条款的数量：意思，vari–
周约，最小，最大，10%，25%，50%，
75%和90%定量

100.–108。学习条款的长度：意思，vari–
周约，最小，最大，10%，25%，50%，
75%和90%定量

Survey Propagation Features

109.–117。调查繁殖的信心：为
每个变量，计算p（true）/
p（false）或p（false）/
p（true）的越高。然后计算跨变量的统计信息
：平均值，变化系数，分钟，最大值，10%，25
%，50%，75%和90%定量

18.–126. **Unconstrained variables:** For each vari-
能够，计算p（无约束）。然后计算跨变量的统
计信息：均值，变化系数，min，max，10%，

25%, 50%, 75%, and 90% quantiles

图2. 11组SAT功能;这些是在[29,27,31]中的。

在ITH目标算法运行中使用的实例的功能。连接参数值，θi和实例特征xi，进入一个输入向量产生训练数据{（[θ1，x1]，o1）。。。，（[θn，xn]，开启）}。根据此数据，我们将学习一个模型，该模型作为输入参数配置θ并在所有培训实例中预测性能。对于GP模型，存在从统计文献中的方法来预测意味着

Problem Size Features:

1.−2。变量和约束的数量：de−

3.线性中的非零条目数
constraint matrix, A

Variable-Constraint Graph Features:

4−7。变量节点度统计：均值，

8−11。约束节点度统计：均值，
max, min, and stddev

Variable Graph (VG) Features:

12−17。节点度统计：MAX，MIN，STDDEV，
25% and 75% quantiles

18−
20.边缘密度：VG中的边数
除以具有相同数量的节点的相同图中的边的边数

LP-Based Features:

21−23。整数松弛矢量：均值，最大，L2

24. LP解决方案的目标函数值

Objective Function Features:

25.标准化系数的标准偏差
$\{c_i/m\}_{i=1}$

26. **Standard deviation of** c_i/n_i ^{n} ，where
ni表示a列中的非零条目的数量

27. $\{c_i/\ \ n_i\}_{i=1}$

Linear Constraint Matrix Features:

28.−29。标准化约束的分布
trix条目，ai，j /
bi：均值和stddev（只有bi = 0的元素）
30.−31。变异系数的标准化系数 −
每行Lute
Nonzero条目：均值和stddev

Variable Type Features:

32.−33。支持离散变量的大小：均值

34.无限性离散变量35%。百分比连续变量百分比

General Problem Type Features:

36.问题类型：分类特征在 −
CPLEX（LP，MILP，固定MILP，QP，MIQP，FIREDMIQP，MIQP，QCP或MIQCP）致敬

37.二次约束数量38.矩阵中的非零条目数量

Q的二次自动目标函数系数，q

39.非零条目的变量数量
Q

图3.混合整数编程问题的八组特征。这些一般的MIP特征已在[11]中作为[32]中的组合获胜者确定问题的特征的概括。

问题实例的性能[21]。但是，由于第4.1节中讨论的问题，当使用日志转换时，此方法不会模拟用户指定的成本度量;例如，而不是算术意味着它将模拟几何平均值。在多个实例的情况下，此问题将特别严重，因为性能通常在一个实例到另一个实例的数量级变化。因此，我们目前没有为多个实例实施SMAC（PP）的版本。相反，我们改编了RF模型来处理多个实例的预测。无论是否参考参数值或实例特征，构造随机林时，所有输入尺寸都同样处理。预测过程如下更改：在每棵树内，我们首先预测给定参数配置和每个实例的组合的性能;接下来，我们将这些预测与用户定义的成本度量（例如，算术平均运行时相结合;最后，我们在树上计算手段和差异。

4.3概括IV：使用模型选择大型混合数/分类配置空间中有希望的配置

SMAC中的“选择配置组件”使用该模型选择有前途参数配置的列表。为了量化配置θ是多么承诺，它使用模型的θ预测分布来计算其预期的正面改进（EI（θ））[15]通过到目前为止所看到的最佳配置（现任者）。EI（θ）对于具有低预测成本的配置θ大，并且对于具有高预测不确定性的成本;因此，它提供了剥削（专注于空间的已知良好部分）和勘探之间的自动权衡（在空间的未知部分收集更多信息）。具体地，我们使用[18]中引入的E [IExp]标准进行对数转换成本;鉴于预测均值μθ和方差σ2

θ的日志变换成本

配置θ，定义为

$$EI(\boldsymbol{\theta}) := E[I_{\exp}(\boldsymbol{\theta})] = f_{min}\Phi(v) - e^{\frac{1}{2}\sigma_{\boldsymbol{\theta}}^2+\mu_\theta} \cdot \Phi(v-\sigma_{\boldsymbol{\theta}}), \tag{3}$$

where $v := \frac{\ln(f}{\Sigma}$ θ，ϕ表示标准的累积分布函数
正态分布，FMIN表示θinc.5的实证平均性能

具有定义的EI（θ），我们仍然决定如何识别配置θ
大ei（θ）。这通常跨参数配置空间的最大化问题。以前的SMBO方法[16,17,18,19]简单地应用于此任务的随机采样（特别是，它们评估了10
000个随机样本的EI），这在高维配置空间中不太可能足够，特别是如果有前途的配置 - 第一个稀疏。要收集一套具有低计算开销的有前途的配置，我们执行一个简单的多启动本地搜索，并考虑与当地最大ei.6的所有产生的配置，这次搜索在精神上与发行方式相似[8,9]，但代替算法性能，它优化EI（θ）（见等式3），可以基于模型预测μθ和σ2来评估

θ而不运行目标ilgo−
rithm。更具体地说，我们本地搜索的细节如下。我们计算以前目标算法中使用的所有配置的EI，从最大ei中选择十个配置，并在每个配置中初始化本地搜索。为了无缝处理混合分类/数字参数空间，我们使用随机的单交换邻域，包括所有配置的集合，这些配置在恰好一个离散参数的值中不同，以及每个数字参数的四个随机邻居。特别地，我们将每个数值参数的范围归一化为[0,1]，然后在具有当前值V的数值参数的四个“相邻”值，从单变量高斯分布，具有平均值V和标准偏差0.2，抑制了新的值间隔[0,1]。由于一组N配置的批量模型预测（以及因此批量EI计算）比N个配置的单独预测便宜得多，因此我们使用最佳改进搜索，一次评估所有邻居的EI;我们一旦邻居都有更大的ei，我们就停止了每个本地搜索。自从

¯5在TB−SPO [19]中，我们使用FMIN =μ（θinc）+σ（θinc）。但是，我们现在相信设置

6我们计划在未来调查更好的机制。但是，我们注意到最好的问题
配方不明显，因为我们渴望具有高EI的多样化配置。

SMBO有时会评估每个迭代的许多配置，因为批量ei
compitations便宜，我们只需计算EI即可增加10
000个随机采样的配置;然后，我们以ei的降序排序所有10
010配置。（本地搜索的十个结果通常比所有随机采样的配置更大。）

选择此列出的10 010配置基于模型，我们交错
随机采样配置，以便为未来模型提供无偏见的培训数据。更确切地说，我们在列表中的配置和随机采样的附加配置之间交替。

4.4 SMAC和ROAR的理论分析

在本节中，我们为有限配置空间提供了SMAC（和咆哮）的收敛性证明。由于强化总是比较对当前现任者的至少两种配置，因此在SMBO的每个迭代中评估至少一个随机采样的配置。因此，在有限配置空间中，每个配置具有在每次迭代中选择的正概率。结合强化增加了用于无法解释每个配置的运行数量的事实，这使我们能够证明SMAC（和咆哮）最终在使用用户定义的成本度量的一致估算器时会聚到最佳配置。证明是直接的，跟随与以前关于聚焦的证据相同的参数（见[9]）。尽管如此，我们在这里给它完整起见。

定义1（一致估计）CN（θ）是C（θ）IFF的一致估计器

$$\forall \epsilon > 0 : \lim_{N \to \infty} P(|\hat{c}_N(\boldsymbol{\theta}) - c(\boldsymbol{\theta})| < \epsilon) = 1.$$

当我们估计基于n运行的参数配置的真实成本c（θ）
是CN（θ），当CN（θ）是C（θ）的一致估计器时，随着N接近无穷大，成本估计变得越来越可靠，最终消除了比较两个参数配置时错误的可能性。以下引理在[9]中是lemma 8;对于证明，请参阅该纸张。

Lemma
2（N→∞的错误）设θ1，θ2≠θ是C（θ1）<c（θ2）的任何两个参数配置。然后，对于一致的估计器CN，LIMN→∞P（CN（θ1）≥CN（θ2））= 0。

仍有待显示的所有是SMAC评估每个参数配置
无限的次数。

LEMMA 3（无界数的评估数）设有n（j，θ）表示运行SMAC的数量在SMBO IT−eration J的末尾的参数配置θ执行。然后，如果SMBO的参数MAXR设置为∞，则为任何常数k和配置θ∈θ（具有有限θ），LIMJ→∞P[n（j，θ）≥k] = 1。

证明。在每个SMBO迭代中，SMAC评估至少一个随机配置（自MAXR
=∞执行至少一个新运行），并且具有p = 1 /
|θ|，这是配置θ。因此，用θ执行的运行的数量由二项式随机变量b（k;
j，p）较低。然后，对于任何常数k <k，我们获得Limj→∞B（k; j，p），因此Limj→∞p
[n（j，θ）≥k] = 1。

当SMAC与MAXR
=ψ基于一致估计器CN和有限配置空间θ优化了成本测量C时，它发现真正的最佳参数配置θ⋆θ θ的概率允许配置进入无限。

证明。每个SMBO迭代都需要有限时间。因此，随着时间的时间，所以SMBO迭代的数量，J.根据引理3，因为j进入无穷大n（θ）对于每个θ ∈ θ来增长。对于每个θ1，θ2，作为n（θ1）和n（θ2）转到无穷大，引理2状态在一对比较中，将优选真正更好的配置。因此，最终，SMAC访问所有有限的参数配置，并将最佳的一个与所有其他参数配置一起接近一个接近概率。

我们注意到，无论使用的模型类型如何，此融合结果都包含。在
事实上，它甚至可以通过参数配置循环循环的简单循环程序。因此，我们依靠经验结果评估SMAC。在解释如何构建相关模型之后，我们在以下部分介绍了这些部分。

5 Experimental Evaluation

我们现在可以针对各种配置方案进行SMAC，ROAR，TB−SPO [19]，GGA
[10]和参数（特别是FOCUSEILS
2.3）[9]的性能，针对各种配置方案，旨在针对解决SAT的算法运行时和mip问题。原则上，我们的咆哮和SMAC方法还适用于优化其他成本指标，例如解决方案质量，算法可以在固定时间预算中实现;我们计划在不久的将来对这种案例进行研究。

5.1 Experimental Setup

配置方案。我们使用了从文献中的17个配置方案，播放了本地搜索SAT求解器SAP的配置[33]（4参数），树搜索求解矛[34]（26参数），以及最广泛使用的商业混合整数编程求解器，IBM ILOG CPLET7（76参数）。
SAP是一种动态的本地搜索算法，它的四个连续参数控制子句权重的缩放和平滑，以及随机步骤的百分比。我们使用其UBCSAT实现[35]。
Spear是用于为工业实例开发的SAT解决方案的树搜索算法，并且具有适当的参数设置，它是某些类型的SAT编码硬件和软件验证实例的最佳可用求解器[13]。矛有26个参数，包括十个分类，四个整数和12个连续参数。分类参数主要控制可变和值选择的启发式，节目排序，分辨率排序，以及启用或禁用优化，例如纯文字规则。连续和整数参数主要处理活动，衰减和消除变量和条款，以及随机重启的间隔和随机选择的百分比。
CPLEX是用于解决MIPS的最广泛使用的商业优化工具，目前使用

[7] http://ibm.com/software/integration/optimization/cplex-optimizer

超过13多家公司和政府机构，以及超过1
000名大学的研究人员。在定义CPLEX的配置空间中，我们小心地保留改变问题制定的所有参数（例如，诸如最优差距的参数，该参数被认为是最佳的解决方案。我们选择的76个参数会影响CPLEX的所有方面。它们包括12个预处理参数（主要是分类）;17
MIP策略参数（主要是分类）;11分类参数决定使用哪种类型的削减程度;9数值MIP“限制”的比例;10个单纯x参数（其中一半分类）;6屏障优化参数（主要是分类）;和11个进一步的参数。大多数参数具有“自动”选项作为其值之一。我们允许这个值，但也包括其他值（分类参数的所有其他值，以及数值参数的一系列值）。

在所有17个配置方案中，我们终止了目标算法在Kmax= 5时运行
秒，以前工作中使用的相同的每次运行截止时间用于这些方案。在以前的工作中，我们还申请了参数，以优化MIP求解器，每次运行非常大的累积时间（最多可达Kmax= 10
000s），并获得比CPLEX调整工具的更好的结果[1]。我们认为，对于如此大的备案，自适应封盖机制，例如在发行方法[9]中实现的自适应封盖机制是必不可少的;我们目前正在努力将这样的机制集成到SMAC.8中，研究SMAC的其余组成部分，我们只使用具有5秒的小纪录的场景。

为了使GGA公平比较，我们改变了优化目标
在原始PAR−10的所有17场景中（惩罚平均运行时，将κmax的超时计数为10・GGA不支持的10・Kmax）到简单的平均运行时（PAR−1，在κmax上计数超时）。9但是，一个差异仍然存在：我们最小化目标算法报告的运行时，但GGA只能最小化其自身测量目标算法运行时，包括在实例中读取的目标算法运行时的测量值。我们使用的所有实例可在http://www.cs.ubc.ca/kabs / beta / projects / aac提供。

参数变换。一些数值参数在非统一尺度上自然变化（例如，具有间隔[100,1600]的参数θ，我们将其离散到参数中使用的值{100,200,400,800,1600}）。我们将这些参数转换为它们更统一地变化的域（例如，日志（θ）∈[log（100），log（1600））），对每个呼叫的对目标算法的参数值进行翻译。

¯8实际上，配置方案Corlat的初步实验（从[1]，kmax =
10
000s）突出显示SMAC的自适应封装机制的重要性：例如，在SMAC的运行中，它仅执行49个目标算法运行，其中15个在κMAX= 10 000秒后定时，另外3个超过5
000每秒秒。这些运行在一起超出了2个CPU天的时间预算（172
800秒），尽管所有这些都可以在不到100秒后安全地切断。结果，对于场景Corlat，SMAC比具有κ= 10
000秒的发行者的参数表现为3倍。另一方面，即使在相对较高的累积时间内，SMAC也可以实现强烈的性能;例如，在κmax= 300s的Corlat上，SMAC优于比例为1.28。

9使用PAR−10比较剩余的配置器，我们的定性结果没有改变。

比较配置过程。我们在每个配置方案上执行了25个每个配置过程。对于每个这样的运行RI，我们计算了测试性能Ti如下。首先，我们在配置过程耗尽其时间预算时提取了现任配置θinc;由于施工和使用模型而导致的SMAC的开销被算作本预算的一部分。接下来，在使用与训练期间相同的每次运行截止时间的离线评估步骤中，我们测量了由θinc参数化的目标算法的1
000个独立测试运行的平均运行时间ti。在多个实例方案的情况下，我们使用了先前未经检验的实例的测试集。对于给定的场景，这导致测试性能T1，。。。
，每个配置过程的T25。我们在这些25个值中报告中位数，可视化其在Boxplots中的方差，并执行Mann−Whitney U测试以检查配置过程之间的显着差异。我们通过HAL
[36]，使用GGA的作者推荐的参数设置，在电子邮件通信中：我们将人口大小设置为70，几代数量为100，运行的数量在第一个生成为5，以及在最后一代中执行的运行数量为70.我们使用了Focusedils
2.3的默认设置，包括积极封盖。我们注意到，在GGA和Focusedils的先前比较[10]中，Cofusedils的封装被禁用;这解释了其在那里的性能不佳，其在这里更好的表现。

除了Focusedils，我们研究的所有配置程序
这里支持数字参数，无需离散化。我们呈现出用于混合数值/分类参数空间这些方法搜索的结果，并启用直接比较与FocusedIls−for完全离散的配置空间。

计算环境。我们在具有2MB缓存和2GB RAM的55个双3.2GHz Intel Xeon
PC上进行了所有实验，运行OpenSUSE Linux
11.1。我们在这些参考机器上测量了运行次数作为CPU时间。

5.2单实例场景的实验结果

为了评估我们的新一般算法配置程序一次咆哮和SMAC一个组件，我们首先评估它们的性能，优化SAP的连续参数和单个SAT实例上的SPEAR的混合数/分类参数;在下一节中研究了多实例方案。要使与我们之前的SMBO Instantiation
TB−Spo进行比较，我们使用[19]中介绍的6个配置方案，该方案旨在将SAPS运行时最小化6个单个SAT编码的实例，3来自Quasigroup完成（QCP
[37]）从小世界图形着色（SWGCP
[38]）。我们还使用了5个类似的新配置方案，该方案旨在优化5个进一步的SAT编码的实例：3从软件验证（SWV [39]）和来自IBM界限模型检查（IBM
[40];我们只使用2个低电平这种硬发行量的定量，因为矛无法在截止时间内的75％位数处解决实例）。配置方案名为算法 −
分布定量：例如，SAPS−QCP−MEDIAM在中位硬质QCP实例上优化SAPS性能。每种算法配置运行的时间预算为30分钟，完全遵循[19]。

基于模型的方法SMAC和TB−SPO在此比较中表现最佳，
其次是咆哮，焦点，和GGA。表1显示了每个配置过程所实现的结果，用于全参数配置空间（包括数字参数）和我们为Focusedils使用的离散化版本。对于单一实例的特殊情况和少数所有数值参数，SMAC（PP）和TB−SPO非常相似，并且都是最好的.10而TB−SPO不适用于剩余的配置方案，我们的更多信息一般SMAC方法在所有这些中实现了最佳性能。对于所有数字参数，SMAC使用PP模型稍微执行稍微更好，而在存在分类参数中，RF模型更好地执行。大多数SAPS场景（4个参数）中，大多数SAPS场景中最好（即，与最佳的最佳或没有明显不同）的最佳（即，最佳或没有显着不同）的咆哮声）。
GGA和Focused都比SAPS情景的咆哮略差略差，略微（但统计学）比SMAC更糟糕，因为大多数矛配置场景。图4可视化每个配置器的25个测试性能，以获取所有方案。我们注意到SMAC和ROAR经常产生比FOCUSEILS和GGA更强大的结果：对于许多情况，其中一些FOCUSELS和GGA运行非常糟糕。

我们的新SMAC和ROAR方法能够探索完整配置
与离散化配置空间加法相比，有时导致显着提高性能的空间限制为。比较表1的右侧左侧的左侧，我们注意到SAPS离散化（相同的是，我们用于优化前面的工作中的Paramils的SAP
[8,9]）在探索全部空间时留下了大量的改进空间：大约1.15−QCP和SWGCP实例分别折叠和1.55倍的加速。GGA并未受益于允许探索SAPS方案的完整配置空间;然而，在其中一个Spear场景（SPEAR−IBM−MED）中，它确实为全部空间执行了1.15倍（尽管仍比SMAC仍然差）更好。

5.3一般多实例配置方案的实验结果

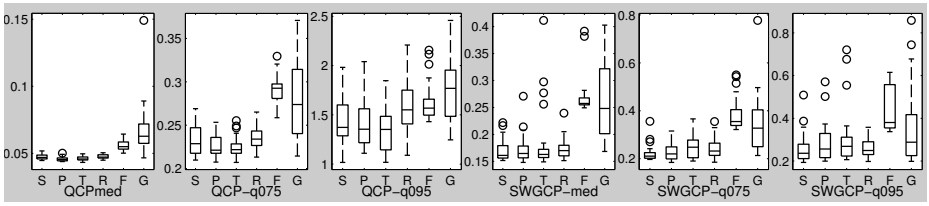
我们现在比较SMAC，Roar，GGA和Focusedils的六个一般算法配置任务的性能，该任务旨在最大限度地减少SAP，SPAR，SPEAR和CPLEX的平均运行时间，用于各种情况。这些是[9]中使用的5个广泛配置方案，以评估发行者的性能，以及另一个CPLEX方案，我们使用每个配置运行的时间预算为5小时。这些实例来自以下域名：quasigroup完成，QCP [37];小世界图形着色，SWGCP [38];组合拍卖中的获胜者确定，地区100
[41];混合整数背包，mik [42]。

总体而言，SMAC在此比较中表现最佳：如表2所示
在所有6个配置场景中，曼数是所有6个配置场景中的最佳（即，统计上无法区分），适用于离散化和完整配置空间。我们简单的咆哮方法令人惊讶地表现出很好，表明了重要性

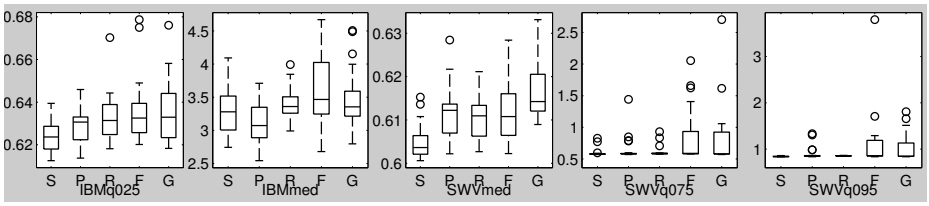
10在这个特殊情况下，两者只有预期的改进标准和它的优化
tion.

Scenario	Unit	Full configuration space					Discretized configuration space				
		SMAC	SMAC(PP)	TB-SPO	ROAR	GGA	SMAC	SMAC(PP)	ROAR	F-ILS	GGA
SAPS-QCP-MED	$[\cdot 10^{-2}s]$	4.70	**4.54**	**4.58**	4.72	6.28	**5.27**	**5.27**	**5.25**	5.50	6.24
SAPS-QCP-Q075	$[\cdot 10^{-1}s]$	**2.29**	**2.21**	**2.22**	2.34	2.74	**2.87**	**2.93**	**2.92**	**2.91**	**2.98**
SAPS-QCP-Q095	$[\cdot 10^{-1}s]$	**1.37**	**1.35**	**1.35**	1.55	1.75	**1.51**	**1.58**	**1.57**	**1.57**	1.95
SAPS-SWGCP-MED	$[\cdot 10^{-1}s]$	**1.61**	**1.65**	**1.63**	**1.7**	2.48	**2.54**	2.59	**2.58**	2.57	2.71
SAPS-SWGCP-Q075	$[\cdot 10^{-1}s]$	**2.11**	**2.2**	**2.48**	2.32	3.19	**3.26**	3.41	3.38	3.55	3.55
SAPS-SWGCP-Q095	$[\cdot 10^{-1}s]$	**2.36**	**2.56**	**2.69**	**2.49**	**3.13**	**3.65**	3.78	**3.79**	**3.75**	**3.77**
SPEAR-IBM-Q025	$[\cdot 10^{-1}s]$	**6.24**	6.31	—	6.31	6.33	**6.21**	6.27	6.30	6.31	6.30
SPEAR-IBM-MED	$[\cdot 10^{0}\ s]$	3.28	**3.07**	—	3.36	3.35	**3.16**	**3.18**	3.38	3.47	3.84
SPEAR-SWV-MED	$[\cdot 10^{-1}s]$	**6.04**	6.12	—	6.11	6.14	**6.05**	6.12	6.14	6.11	6.15
SPEAR-SWV-Q075	$[\cdot 10^{-1}s]$	**5.76**	5.83	—	5.88	5.83	**5.76**	5.89	5.89	5.88	5.84
SPEAR-SWV-Q095	$[\cdot 10^{-1}s]$	**8.38**	8.5	—	8.55	8.47	**8.42**	8.51	8.53	8.58	8.49

表1.算法配置过程的比较，用于优化单个问题实例的参数。我们执行了25个每个配置过程的独立运行，并报告25个测试性能的中位数（跨越1000个目标算法的平均运行时间使用找到的配置运行）。我们基于Mann−Whitney
U测试，我们基于各个配置空间的最佳配置器的配置器的粗体指南的配置。符号“ − ”表示配置器不适用于此配置空间。



(a) SAPS (4 continuous parameters)



（b）矛（26个参数;其中12个连续和4个积分）

图4.配置过程对单个实例设置SAP和SPEAR参数的性能的视觉比较。对于每个配置器和方案，我们向下表1的25个测试性能显示盒子，用于完整配置空间（对于Focusedils离散化）。对于SMAC，'PP）的SMAC（pp），'t'为tb−spo，'r'为roar，'ffor focuseils，'g'为gga。

强化机制：它是一个配置空间的6个配置场景中的2个中最好的。然而，它表现得比使用最大配置空间的CPLEX −
算法的最佳方法差。我们注意到咆哮的随机采样方法缺乏Focusion的本地搜索或SMAC响应面模型提供的指导。GGA执行了

Scenario	Unit	Full configuration space				Discretized configuration space			
		SMAC	ROAR	F-ILS	GGA	SMAC	ROAR	F-ILS	GGA
SAPS-QCP	$[\cdot 10^{-1}s]$	**7.05**	7.52	—	7.84	**7.65**	**7.65**	**7.62**	**7.59**
SAPS-SWGCP	$[\cdot 10^{-1}s]$	**1.77**	**1.8**	—	2.82	**2.94**	3.01	**2.91**	3.04
SPEAR-QCP	$[\cdot 10^{-1}s]$	**1.65**	1.84	—	2.21	**1.93**	2.01	2.08	2.01
SPEAR-SWGCP	$[\cdot 10^{0}\ s]$	**1.16**	**1.16**	—	1.17	**1.16**	**1.16**	1.18	1.18
CPLEX-REGIONS100	$[\cdot 10^{-1}s]$	**3.45**	6.67	—	4.37	**3.50**	7.23	**3.23**	3.98
CPLEX-MIK	$[\ 10^{0}\ s]$	**1.20**	2.81	—	3.42	**1.24**	3.11	2.71	3.32

表2.算法配置程序对多个阶段的基准的比较。我们执行了25个每个配置过程的独立运行，并报告25个测试性能的中位数（跨越1
000个目标算法的平均运行时间运行，并在训练集中的测试集中的发现配置中运行）。基于Mann−Whitney U测试，我们对配置器的大胆面对配置器的表现不显着更差。

略微优化CPLEX而不是咆哮，但也明显差，而不是聚焦或SMAC。图5可视化所有6个方案实现的每个配置器的性能。我们注意到 – 类似于单一实例的情况 –
SMAC的结果通常比聚焦和GGA更强大。

虽然我们的新方法实现的性能改进可能不是
绝对术语看起来很大，重要的是要记住算法配置是一个优化问题，并且梳理出后几个改善百分比的能力往往区分良好的算法。我们预计配置过程之间的差异在具有较大的每个实例运行时更清晰。为了有效处理这种情景，我们认为SMAC将需要类似于我们为发行方法引入的自适应覆盖机制[9];我们正在积极努力与SMAC的模型集成这样的机制。

如在单一实例的情况下，对于某些配置方案，SMAC和ROAR
当允许探索全部空间而不是聚焦 –
ILS的离散搜索空间时，取得了更好的结果。SAP的加速与单一实例案例中观察到的SAP（用于SAPS−QCP的约1.15倍，对于SAPS−SWGCP为1.65倍），但现在我们也观察到SPEAR−QCP的1.17倍改善。相比之下，在允许探索全部空间时，GGA实际上在6个方案中的4个方案进行了更糟。

6 Conclusion

在本文中，我们在顺序模型的优化（SMBO）上扩展了前一行，以解决一般算法配置问题。SMBO先前仅应用于单个问题实例上具有数值参数的算法的优化。我们的工作克服了这两个限制，允许对问题实例集的分类参数和配置。这一技术的四种技术进步是（1）一种新的强化机制，在配置之间采用堵塞的比较;替代类响应面模型，随机林，处理（2）分类参数和（3）多个实例;（4）新的优化过程，以在大的混合分类/数字空间中选择最有前途的参数配置。

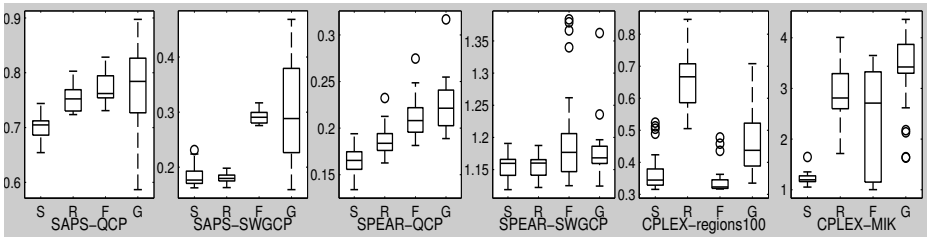


图5.通用算法配置SCE-NARIOS配置过程的视觉比较。对于每个配置器和方案，我们向底板显示底板表2的Boxplots，用于完整配置空间（对于FocusedIls离散化）。's'代表SMAC，'r'for roar，'f'for focusedils，而且为gga'g'。Focusedils不适用于完整配置空间，由“-”表示。

我们为两个SAT算法的配置提出了经验结果（一个本地搜索，一个树搜索）和商业MIP求解器CPLEX总共有17个配置场景，展示了我们方法的强度。总的来说，我们的新SMBO程序SMAC产生了统计上显着的 - 虽然有时会对几种配置方案的所有其他方法进行小，但从未表现差。与Focusedils相比，我们的新方法也能够搜索完整（非离散化）配置空间，这导致了几种配置方案的进一步改进。我们注意到，我们的新强化机制甚至可以咆哮，一种简单的无模型方法，在许多情况下比以前的通用配置程序更好;咆哮仅对优化CPLEX进行速度，良好的配置稀疏。SMAC在咆哮中进一步改进，最重要的是 - 最先进的CPLEX配置的最新性能。

在未来的工作中，我们计划改进SMAC以更好地处理配置方案对于每个目标算法运行，具有大的每次运行累积时间;具体而言，我们计划将Paramils的自适应封装机制集成到SMAC中，这将需要扩展SMAC模型来处理所产生的部分删除的数据。虽然在本文中，我们旨在找到一个具有整体良好性能的单一配置，我们还计划使用SMAC的模型来确定每个实例的良好配置。最后，我们计划使用这些模型来表征各个参数及其交互的重要性，并在参数和实例功能之间研究交互。

Acknowledgements

我们感谢克文墨菲对这项工作的建模方面进行了许多有用的讨论。还要感谢Chris Fawcett和Chris Nell，提供GGA通过HAL运行GGA，以及凯文蒂尔尼以获得GGA的参数，以及詹姆斯风格和Mauro Vallati对本文的早期汇票进行评论。FH感谢加拿大国际局国际博览会研究奖学金的支持

教育。HH和KLB通过各自的发现拨款，并通过种子项目补助金感谢NSERC的支持。

A Proof of Theorem 1

为了证明kernel函数kmixed的有效性，我们首先证明了以下引理：

对于任何有限域θ，核函数KHAM：θ × θ→R定义为

$$K_{ham}(x,z) = \exp(-\lambda \cdot [1 - \delta(x,z)]) \tag{4}$$

是一个有效的内核功能。

证明。我们使用任何常量是有效内核功能的事实，并且在加法和乘法下关闭内核功能的空间。如果我们可以找到嵌入ϕ，例如k（x，z）=ϕ（x）t·ϕ（z）[43]，我们还使用内核函数k（x，z）有效的事实是有效的。

让A1，。。。，我表示是θi的最多元素，以及每个元素ai
定义仅包含零的M维指示器向量VAI，除了在位置I，它是一个。定义X，Z≥θi的内核函数K1（x，z）作为M维空间中嵌入式ϕ（x）和ϕ（z）的点乘积：

$$k_1(x,z) = \boldsymbol{v}_x^{\intercal} \cdot \boldsymbol{v}_z = \sum_{i=1}^{m} \boldsymbol{v}_x(i) \cdot \boldsymbol{v}_z(i) = 1 - \delta(x,z).$$

要以公式4的形式带来这一点，我们添加了恒定的内核函数

$$k_2(x,z) = c = \frac{\exp(-\lambda)}{1-\exp(-\lambda)},$$

然后乘以常量内核函数

$$k_3(x,z) = 1/(1+c) = 1 - \exp(-\lambda).$$

因此，我们可以将函数KHAM重写为有效内核的产品，从而证明其有效性：

$$\begin{aligned} K_{ham}(x,z) &= (k_1(x,z) + k_2(x,z)) \cdot k_3(x,z) \\ &= \begin{cases} 1 & \textit{if } x = z \\ \exp(-\lambda) & \textit{otherwise} \end{cases} \\ &= \exp\left[-\lambda\delta(x \neq z)\right]. \end{aligned}$$

定理1（kmixed的有效性）。内核kmixed：θ × θ→r定义为

$$K_{mixed}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \exp\left[\sum_{l \in \mathcal{P}_{cont}} (-\lambda_l \cdot (\theta_{i,l} - \theta_{j,l})^2) + \sum_{l \in \mathcal{P}_{cat}} (-\lambda_l \cdot [1 - \delta(\theta_{i,l}, \theta_{j,l})])\right].$$

是任意配置空间的有效内核功能。

证明。由于kicixed是有效的高斯内核的产品

$$K(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \exp\left[\sum_{l \in \mathcal{P}_{cont}} (-\lambda_l \cdot (\theta_{i,l} - \theta_{j,l})^2)\right]$$

每个分类参数的一个kham内核，它是一个有效的内核。

References

[1] F. Hutt，H. H.Haos和K.Leyton−Brown。混合整数的自动配置
编程求解器。在proc。CPAIOR−10，第186−202页，2010年。
[2] S. Minton，M. D. Johnston，A. B. Philips和P. Laird。最大限度地减少冲突：启发式
约束满意度和调度问题的修复方法。AIJ，58（1）：161−205,1992。

[3] J. Greatch和G. dejong。作曲家：实用程序问题的概率解决方案
加速学习。在proc。AAAI−92，第235−240页，1992年。
[4] S.P.Coy，B. L. Golden，G. C.跑器和E. A. Wisil。使用实验设计寻找
启发式的有效参数设置。HeuRistics，7（1）：77−97，2001。
[5] B. Adenso−Diaz和M. Laguna。使用分数实验进行微调算法
设计和本地搜索。运营研究，54（1）：99−114，2006年1月至2月。
[6] P. Balaprakash，M. Birattari和T.St utzle。F竞赛算法的改进策略：
采样设计和迭代细化。在proc。MH−07，页面108−122,2007。
[7] M. Birattari，Z. Yuan，P. Balaprakash和T.St ¨ utzle。分析的实证方法
优化算法，F族和迭代F竞赛：概述。Springer，柏林，德国，2010年。

[8] F. Hutter，H. H.Hoos和T.St ¨ utzle。基于本地的自动算法配置
搜索。在proc。AAI −77，PIGO 1122−257，2007。
[9] F. Hutt，H.H.Haos，K.Leyton−Brown和T.St ¨ utzle。发行方法：自动算法
configuration framework. *JAIR*, 36:267–306, October 2009.
[10] C. Ans'otegui，M. Sellmann和K. Tierney。基于性别的遗传算法
自动配置算法。在proc。CP−09，第142−157页，2009年。
[11] F. Hutter。解决硬计算问题的算法自动配置。
博士论文，不列颠哥伦比亚省大学，加拿大温哥华，2009年。
[12] M. Birattari，T.St ¨ utzle，L. paquete和K.Varrentropp。一种配置的赛车算法
陨石学。在proc。Gecco−02，第11−18页，2002年。
[13] F. Hutter，D. Babi'c，H. H. Hoos和A. J. Hu。通过自动调整验证
决策程序。在proc。FMCAD'07，第27−34页，2007年。
[14] A. KhudaBukhsh, L. Xu, H. H. Hoos, and K. Leyton-Brown. SATenstein: Automatically
从组件构建本地搜索SAT求解器。在proc。IJCAI−09,2009。
[15] D. R. Jones，M. Schonlau和W. J. Welch。高效的全球优化昂贵的黑色
盒子功能。全球优化杂志，13：455−492,1998。
[16] T. Bartz−Beielstein，C.Lasarczyk和M. Preuss。顺序参数优化。在
Proc。CEC−05，第773−780页。IEEE按，2005年。
[17] T. Bartz−Beielstein。进化计算的实验研究：新实验 −
心理学。自然计算系列。Springer Verlag，柏林，2006年。
[18] F.Hutt，H.H.Haos，K.Leyton−Brown和K.P.Murphy。实验调查
基于模型的参数优化：SPO及以后。在proc。Gecco−09,2009。
[19] F. Hutch，H.H.Haos，K.Leyton−Brown和K.P.Murphy。时间有限的顺序
参数优化。在proc。狮子4，第281−298页，2010年第281−298页。

[20] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
[21] B. J. Williams, T. J. Santner, and W. I. Notz. Sequential design of computer experiments to minimize integrated response functions. *Statistica Sinica*, 10:1133–1152, 2000.
[22] D. Huang, T. T. Allen, W. I. Notz, and N. Zeng. Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of Global Optimization*, 34(3):441–466, 2006.
[23] Julia Couto. Kernel k-means for categorical data. In *Advances in Intelligent Data Analysis VI (IDA-05)*, volume 3646 of *LNCS*, pages 46–56, 2005.
[24] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
[25] T. Bartz-Beielstein and S. Markon. Tuning search algorithms for real-world applications: A regression tree based approach. In *Proc. of CEC-04*, pages 1111–1118, 2004.
[26] M. Baz, B. Hunsaker, P. Brooks, and A. Gosavi. Automated tuning of optimization software parameters. Technical Report TR2007-7, Univ. of Pittsburgh, Industrial Engineering, 2007.
[27] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. SATzilla: portfolio-based algorithm selection for SAT. *JAIR*, 32:565–606, June 2008.
[28] K. Leyton-Brown, E. Nudelman, and Y. Shoham. Empirical hardness models: Methodology and a case study on combinatorial auctions. *Journal of the ACM*, 56(4):1–52, 2009.
[29] E. Nudelman, K. Leyton-Brown, H. H. Hoos, A. Devkar, and Y. Shoham. Understanding random SAT: Beyond the clauses-to-variables ratio. In *Proc. of CP-04*, 2004.
[30] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer Verlag, 2nd edition, 2009.
[31] L. Xu, F. Hutter, H. Hoos, and K. Leyton-Brown. SATzilla2009: an automatic algorithm portfolio for sat. Solver description, SAT competition 2009, 2009.
[32] K. Leyton-Brown, E. Nudelman, and Y. Shoham. Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In *Proc. of CP-02*, pages 556–572, 2002.
[33] F. Hutter, D. A. D. Tompkins, and H. H. Hoos. Scaling and probabilistic smoothing: Efficient dynamic local search for SAT. In *Proc. of CP-02*, pages 233–248, 2002.
[34] D. Babić. *Exploiting Structure for Scalable Software Verification*. PhD thesis, University of British Columbia, Vancouver, Canada, 2008.
[35] D. A. D. Tompkins and H. H. Hoos. UBCSAT: An implementation and experimentation environment for SLS algorithms for SAT & MAX-SAT. In *Proc. of SAT-04*, pages 306–320, 2004.
[36] C. Nell, C. Fawcett, H. H. Hoos, and K. Leyton-Brown. HAL: A framework for the automated analysis and design of high-performance algorithms. In *LION-5*, page (to appear), 2011.
[37] C. P. Gomes and B. Selman. Problem structure in the presence of perturbations. In *Proc. of AAAI-97*, pages 221–226, 1997.
[38] I. P. Gent, H. H. Hoos, P. Prosser, and T. Walsh. Morphing: Combining structure and randomness. In *Proc. of AAAI-99*, pages 654–660, 1999.
[39] D. Babić and A. J. Hu. Structural Abstraction of Software Verification Conditions. In *Proc. of CAV-07*, pages 366–378, 2007.
[40] E. Zarpas. Benchmarking SAT Solvers for Bounded Model Checking. In *Proc. of SAT-05*, pages 340–354, 2005.
[41] K. Leyton-Brown, M. Pearson, and Y. Shoham. Towards a universal test suite for combinatorial auction algorithms. In *Proc. of EC'00*, pages 66–76, New York, NY, USA, 2000. ACM.
[42] A. Atamtürk. On the facets of the mixed–integer knapsack polyhedron. *Mathematical Programming*, 98:145–175, 2003.
[43] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.