由一个指令核组成的多核系统的可靠性增强

Shashikiran Venkatesha [#, a]，Ranjani Parthasarathi [b]

[a] 信息科学与技术系研究学者，工程学院

金迪，安娜大学，金奈 600025，泰米尔纳德邦，印度。

[b] 教授，工程学院金迪校区信息科学与技术系

安娜大学，钦奈 600025，泰米尔纳德邦，印度。

摘要

CMOS器件尺寸的快速缩小使单个芯片上集成了数十亿个晶体管，并促成了多内核集成，同时带来了诸多挑战，例如功耗增加、热耗散加剧，以及瞬态故障和永久性故障的发生。在多核场景下，缓解内核级别的瞬态故障和永久性故障已成为重要的设计参数。内核级技术是一种基于冗余的故障缓解技术，可提高多核系统的生命周期可靠性。在非对称多核系统中，由较小内核为较大内核提供容错能力，是一种日益受到众多研究人员关注的内核级故障缓解技术。本文提出了一种经济型非对称多核系统，其中包含一个指令核（MCS-OIC）。“硬件成本估计”指对MCS-OIC的功耗和面积进行估计。在MCS-OIC中，OIC是一个热备冗余内核。OIC可在较短时间内为传统核心提供功能支持。为评估这一构想，本文使用现场可编程门阵列和ASIC对MCS-OIC的不同配置进行综合。最大功耗开销和最大面积开销分别为0.46%和11.4%。本文采用单次系统（OSS）模型对MCS-OIC中OIC的行为进行建模，以开展可靠性分析。OSS的模型参数，即准备度、唤醒概率和启动策略，被映射到包含OIC的多核系统中。本文推导了系统可靠性表达式，并针对特殊情形估算了系统可靠性。

关键词：可靠性、多核、一指令核、一次性系统、冗余分配问题、遗传算法和粒子群优化。

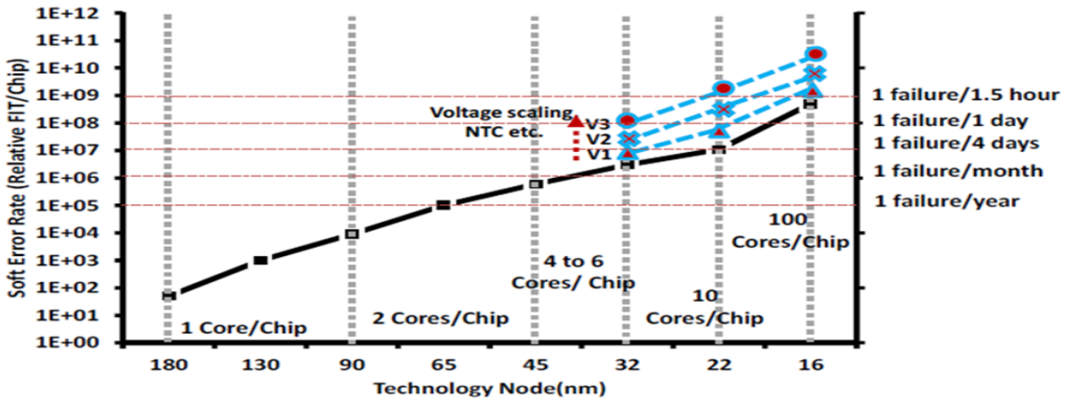
1. 引言

芯片上的晶体管数量一直每 18 个月翻一番，正如摩尔定律所预测的那样[1,2]。单核的性能扩展遵循了摩尔定律所确立的发展轨迹。晶体管密度每年增加约 35%，裸片尺寸每年增加 10% 至 20% [3]。尽管取得了这些进步，每瓦性能/每平方毫米裸片面积的性能却在下降。随着晶体管尺寸缩小，功率密度随之增加，从而形成了“功耗墙”，将处理器频率限制在 2006 [4] 的 4 GHz。为应对功耗墙，芯片设计人员和芯片制造商开始专注于多核处理器 [5]，并将其扩展为搭配更小型的专用处理器。此后，发展趋势一直是增加内核数量，进而增加芯片上的晶体管数量。

当前版本是版本 1.0。 稿件正在审稿。修改或扩展后的版本将在期刊中发表。

快速集成和高晶体管密度也增加了内核的脆弱性 [6]。更快的 CMOS 集成带来了诸多挑战，包括功率/热耗散增加、电路故障频发以及可靠性不足 [7]。因此，亟需采用故障缓解和容错技术。本文提出了一种适用于多核场景的容错解决方案。

故障大致可分为三类：永久性故障或硬故障、间歇性故障和瞬态故障 [8]。电迁移、应力迁移、热循环、时间相关介质击穿和负偏压温度不稳定性会导致永久性故障。氧化层退化、制造残留物和工艺变化会导致间歇性故障。间歇性故障会在一段时间内不规则且频繁地发生。瞬态故障或软错误由宇宙射线、α粒子、空气污染、湿度、噪声和异常电压波动引起 [9]。研究人员预测，每一代技术中，每个逻辑状态位的软错误率将增加约百分之八 [10]。根据《国际器件与系统路线图》（IRDS）2017 年路线图，器件缩放将触及物理极限，故障率将达到每小时 1 次。从图 1.1 可以看出，在 16纳米工艺节点尺寸下，拥有 100 个内核的芯片可能每小时因软错误发生一次故障 [11,12]。



（来源：Shivakumar 等人 2002[11]）。

图 1.1 不同技术节点下的 SER 率

如图1.1 所示，随着每个芯片上的内核数量增加，每个芯片的单位时间故障数（FIT）总量也随之增加。为了容纳更多的每芯片内核数量，需要：（1）保持每个芯片的 FIT 总量不变（或无变化）；（2）降低每个内核的软错误率。在当前的处理器内核中，内核的前端由解码队列、指令转换后备缓冲器和锁存器组成。内核的后端由算术逻辑单元、寄存器文件、数据转换后备缓冲器、重排序缓冲区、内存顺序缓冲区和发射队列组成。内核后端和前端的软错误率分别为 74.48%和 25.22%。在当前的处理器内核中，锁存器经过加固，[13,14],高速缓存和大型存储器阵列则使用纠错码（ECC）进行保护[15]。与处理器前端相比，处理器后端的软错误率更高，这主要是由算术逻辑单元导致的。

处理器内核中算术逻辑单元的 FIT 已开始达到较高水平，因此当前和未来的处理器需要采用稳健的故障缓解方法。因此，解决内核（后端的算术逻辑单元）的可靠性问题，对于提高多核系统的可靠性更为重要 [16,17]。传统的软错误处理方法会消耗更多功耗和面积。因此，本文提出了一种异构模型，采用“低成本”（功耗更低、面积更小）的容错内核，以提高多核系统的可靠性。

我们介绍一种异构多核系统，其中传统核心由单指令核（OIC）[18]提供支持。我们设计了一个 32 位 OIC，为采用 32 位整数指令的传统 MIPS 核心组成的多核系统提供容错能力。OIC 只执行一条指令，即“subleq——小于或等于时相减”。OIC 配备三个传统减法器和一个额外的自检减法器。检测到某个功能单元（即算术逻辑单元）发生故障的传统核心，会将操作码及操作数发送给 OIC。OIC 通过反复执行 subleq 指令来模拟该指令，

当前版本为版本 1.0。 稿件正在审稿。修改或扩展后的版本将在期刊上发表。

以预先确定的方式。我们早期的工作仅关注 OIC 的面积、功耗和可靠性分析。在本文中，我们提出了一种集成解决方案，即对由OIC 支持的传统核心（MIPS 核心）组成的多核系统进行面积、功耗和可靠性分析，这是我们早期工作的一部分 [45]。

OIC是一种备用冗余核心，为传统核心提供功能支持。从图1.1可以观察到，在90纳米工艺下，内核每年会遇到一次软错误，而在16纳米工艺下，软错误频率会上升到每小时一次。这表明，OIC为传统核心提供间歇性支持即可。因此，传统核心所需的OIC支持仅需持续较短的时间。本文提出的可靠性模型刻画了OIC在较短时间区间内为传统核心的算术逻辑单元提供的计算支持。采用单次系统（OSS）这一模型对使用OIC的多核系统进行建模。在OSS中，系统预计仅执行一个或多个任务一次。任务可以重复执行，但与先前的任务执行相互独立。本文提出了基于OSS的可靠性模型、模型参数，以及使用OIC的多核系统（简称MCS-OIC）的系统可靠性估计。

在 MCS-OIC 中，当存在许多传统核心和 OIC 时，对传统核心所需的 OIC 及其功能进行选择 会成为决定系统可靠性的重要考量因素。这种对 OIC 及其功能的选择被视为冗余分配问题（RAP）。此外，还需考虑的另一个因素是，将指令从传统核心迁移到 OIC 并在 OIC 上执行相应功能的可靠性 r(t) 具有不精确性。这种不精确性可能源于制造复杂性、设计复杂性以及部署环境的复杂性。为了解决不精确性，通常有三种可用方法，即区间分析、模糊方法和随机模糊方法。其中，采用区间分析来处理不精确性。因此，为 r(t) 分配区间值可靠性。由此，该 RAP 被称为区间 RAP。本文提出了区间 RAP 的建模方法，并使用改进的遗传算法（GA）和粒子群优化（PSO）进行求解，以处理区间数，这是我们早期 [46] 工作的一部分。

我们总结了各项贡献，并在下文中逐一列出。

(a) MCS-OIC 的功耗和面积分析。

(b) 用于估计 MCS-OIC 系统可靠性的基于 OSS 的可靠性模型。

(c) 使用所提出的遗传算法和粒子群优化对区间数进行区间 RAP 建模，并求解系统可靠性。

本文其余部分组织如下：多核场景下备用冗余系统可靠性建模的相关工作见“2. 备用冗余系统可靠性建模背景”一节；区间数的算术与序关系见“3. 区间分析背景”一节； MCS-OIC（由一个 MIPS 核心和一个 OIC 组成）的微体系结构，以及功耗和面积的仿真结果见“4. MCS-OIC 的微体系结构与实验结果”一节；OSS模型、模型参数、MCS-OIC 的不同配置以及 MCS-OIC 的 OSS 模型参数映射见“5. MCS-OIC 的 OSS 与模型参数”一节；MCS-OIC 的可靠性建模假设、符号说明以及系统可靠性表达式的推导见“6. MCS-OIC 的系统可靠性”一节；RAP、区间 RAP 及其固有特性的讨论见“7. 区间 RAP 及其固有特性”一节；采用所提出的遗传算法（含敏感性分析）和粒子群优化求解区间 RAP 的方法，分别见“8. 区间 RAP 的遗传算法”和“9. 区间 RAP 的粒子群优化”两节；本文结论见“10. 结论”一节。

2. Background on Reliability modelling for standby redundancy systems

传统上，经典二状态串联系统和并联系统具有能够替换故障组件的待机冗余组件。二状态系统可以处于两种状态：成功状态和故障状态。二状态 k-选-n系统在抽象基于待机冗余的容错串联／并联系统方面很受欢迎。如果且仅当 n个组件中至少有k 个组件工作时，一个由 n个组件组成的系统才工作（或“处于良好状态”），这种系统称为k-选-n系统：G 系统，由 Birnbaum 等人引入。[19] 如果且仅当 n 个组件中至少有 k个组件发生故障时，一个由 n个组件组成的系统才失效，这种系统称为 k-选-n系统：F系统。多状态是一个 k-选-n 系统：G 系统，可靠性 R（k、n 和 j）是 k

当前版本为版本 1.0。 稿件正在审稿。修改或扩展后的版本将在期刊上发表。

组件处于高性能状态 (j) 或更高状态。类似地，如果至少有 k 个组件低于低性能状态 (l)，则这种由 n 个组件组成的系统称为 k-out-of-n 系统：F 系统。

关于热备用系统可靠性建模的开创性研究 [20] 探讨了 k-out-of-n 系统（包括串联和并联结构）的平均剩余寿命。指数分布中的失效时间[21], 正态分布 [22] 被用于分析备用冗余系统。Gnedenko 等人 [23] 基于等效年龄的概念，使用一般失效时间分布研究了具有相同组件的 1-out-of-n 温备用系统。Li 等人 [24] 研究了温备用系统中组件的危险率。[25]讨论了 k-out-of-n 温备用系统的可靠性特征。Huang 和 Xu [26] 为同构多核系统提出了三种方案：(a) 备用冗余系统（k-out-of-n 系统：G 系统），(b) 平缓降级系统 (GDS)（所有 n 个内核均为活动内核），以及 (c) 处理器轮换方案。该方法使用 Weibull 分布估算系统可靠性。实验结果表明，随着内核数量增加，系统寿命得以延长。

在上述所有方法中，k 个内核都是备用内核，在活跃内核发生故障时启用。MCS-OIC 中的 OIC 为其他部分活跃的传统核心提供功能冗余，但持续时间较短。组件级冗余和功能级冗余是 MCS-OIC 的关键特征，而传统的 k-out-of-n 系统无法对这些特征进行建模。因此，基于 k-out-of-n 系统的可靠性建模并不适用于包含 OIC 的多核系统。

如上所述，单次系统（OSS）是能够体现 MCS-OIC 中 OIC 所提供功能支持的一种模型。 OSS 曾用于对核武器系统 [27], 、自组传感器网络 [28], 和导弹系统 [29]进行建模。用于自组传感器网络的 OSS 模型经过全面修改，并适配于 MCS-OIC。在 MCS-OIC 中，单次系统（ OSS ）在两种情况下能够合理地抽象 OIC 提供的功能冗余，具体如下：

(a) 当内核受到软错误影响时，OIC 为传统核心提供功能支持，同时内核随后会恢复到稳定状态。

(b) 在传统核心上具有高失败概率的指令执行可以迁移到 OIC。只有这些指令在 OIC 上执行。

第 5 节介绍了 MCS–OIC 的 OSS 建模。

3. 区间分析背景

区间分析最初由[30] 引入，以解决不精确性问题。区间算术与序关系对于遗传算法中寻找更优解的交叉/变异和选择过程非常重要。在粒子群优化中，序关系用于更新 G_{best}、$L_{best,\ i}$ 和 $P_{best,\ i}$ 参数，而区间算术用于更新速度。区间算术与序关系将在下文讨论。区间分析用于解决电力系统[31], 微电子电路 [32], 结构安全分析 [33],和损伤识别 [34]中的不精确性。Wang等人 [35] 提出了基于备用冗余的系统优化问题的区间分析方法。Bhunia和Samanta [36] 提出了区间域中理论多目标优化问题的区间分析方法。Gupta 和 Bhunia [37], 使用了串联系统中组件的区间可靠性。

3.1. 区间数的算术

考虑 W为一个区间值数。它被定义为满足以下条件的实数‘a’的集合，$w_L \le a \le w_R$, $i.e.$, $a \in [w_L, w_R]$, $w_L, w_R \in R$.R是实数集合。w_L 和 w_R 分别是该区间的左边界和右边界。W 在方程 (1) 中表示。

$$W = [w_L, w_R] = \{ a : w_L \le a \le w_R , a \in R \} \quad (1)$$

ed 在方程中间值数字 W 也可以用半径和中心来表示，并且是
(2) 6.2.

当前版本为版本 1.0。 稿件正在审稿。修改版或扩展版将在期刊上发表。

$$W = < w_c, w_r > = \{ a : w_c - w_r \leq a \leq w_c + w_r, a \in R \} \quad (2)$$

$w_c = (w_L + w_R)/2$ 和 $w_r = (w_R - w_L)/2$ 分别是区间的中心和半径，R 是实数集。

定义 1.1：算术运算：设 $X = [X_L, X_R]$ 和 $Y = [Y_L, Y_R]$ 为闭区间。那么加法、减法、乘法和除法定义如下。

两个区间 X 和 Y 的加法如方程 (3) 所示。

$$X + Y = [X_L + Y_L, X_R + Y_R] \quad (3)$$

两个区间的减法以及 Y 如方程 (4) 所示。

$$X - Y = [X_L - Y_L, X_R - Y_R] \quad (4)$$

区间与实数 λ 的乘法由方程 (5) 和 (6) 给出。

$$\lambda . X = \lambda . [X_L, X_R] = [\lambda X_L, \lambda X_R] \ if \ \lambda \geq 0 \quad (5)$$

$$\lambda . X = \lambda . [X_L, X_R] = [\lambda X_R, \lambda X_L] \ if \ \lambda < 0 \quad (6)$$

两个区间的乘法由方程（7）给出。

$$X \text{ x } Y = [\min(X_L Y_L, X_L Y_R, X_R Y_L, X_R Y_R), \max(X_L Y_L, X_L Y_R, X_R Y_L, X_R Y_R)] \quad (7)$$

区间 Y 除以区间 X 的除法由方程（8）给出。

$$\frac{Y}{X} = Y \text{ x } \frac{1}{X} = [Y_L, Y_R] \text{ x } [\frac{1}{X_L}, \frac{1}{X_R}] \text{ provided } X_L \neq 0, X_R \neq 0. \quad (8)$$

定义 1.2：区间的幂：考虑 X =$[X_L, X_R]$。对于非负整数 N，区间 X 的 N^{th} 次幂由方程 (9) 给出。

$$X^N = [1,1], \quad if \ N = 0.$$

$$X^N = [X_L^N, X_R^N], if \ X_L \geq 0, or \ if \ N \ is \ odd$$

$$X^N = [X_R^N, X_L^N], if \ X_R \leq 0, or \ if \ N \ is \ even$$

$$X^N = [0, \max(X_L^N, X_R^N)], if \ X_L \leq 0 \leq X_R, N > 0 \ is \ even. \quad (9)$$

3.2. 区间数的序关系

当前版本是版本 1.0。 稿件正在审稿中。修改版或扩展版将刊登在期刊中。

区间值数对之间的序关系取决于决策者对优化问题的认知。本节将讨论如何为最大化问题从区间值数对中选择最佳备选方案。

T两个区间数 X= $[X_L, X_R]$ 和 Y = $[Y_L, Y_R]$ 可分为以下三种类型 s.

(i) 两个区间互不相交 (ii) 部分重叠的区间

(iii) 一个区间包含在另一个区间内。

Mahato 和 Bhunia [38] 针对最大化问题中的序关系，从乐观和悲观决策者的角度，为这些情况提出了定义。

乐观型决策：

定义 1.3. 最大化问题中 X 与 Y 之间的序关系 $\geq_{o\,max}$ 由不等式 6.10 和（11）给出。

$$X \geq_{o\,max} Y \; iff \; X_R \geq Y_R \tag{10}$$

$$X >_{o\,max} Y \; iff \; X \geq_{o\,max} Y \; and \; X \neq Y. \tag{11}$$

乐观型决策者接受优于 Y 的区间 X。序关系 $\geq_{o\,max}$具有对称性，但不具有传递性。

悲观决策：

定义 1.4. 从悲观型决策者的角度来看，最大化问题中区间 X = $[X_L, X_R]$ = $\langle X_c, X_r \rangle$ 与 Y = $[Y_L, Y_R]$ = $\langle Y_c, Y_r \rangle$ 之间的序关系 $\geq_{o\,max}$ 由不等式（12）和（13）给出。

$$X >_{p\,max} Y \; iff \; X_c > Y_c \; for \; \text{disjoint} \; and \; partially \; ovelapped \; intervals. \tag{12}$$

$$X >_{p\,max} Y \; iff \; X_c \geq Y_c \; and \; X_r < Y_r \; when \; one \; interval \; is \; contained \; in \; the \; other. \tag{13}$$

(6.13)

针对$X_c > Y_c \; and \; X_r > Y_r$，可能需要考虑乐观型决策。

4. MCS-OIC 的微体系结构与实验结果

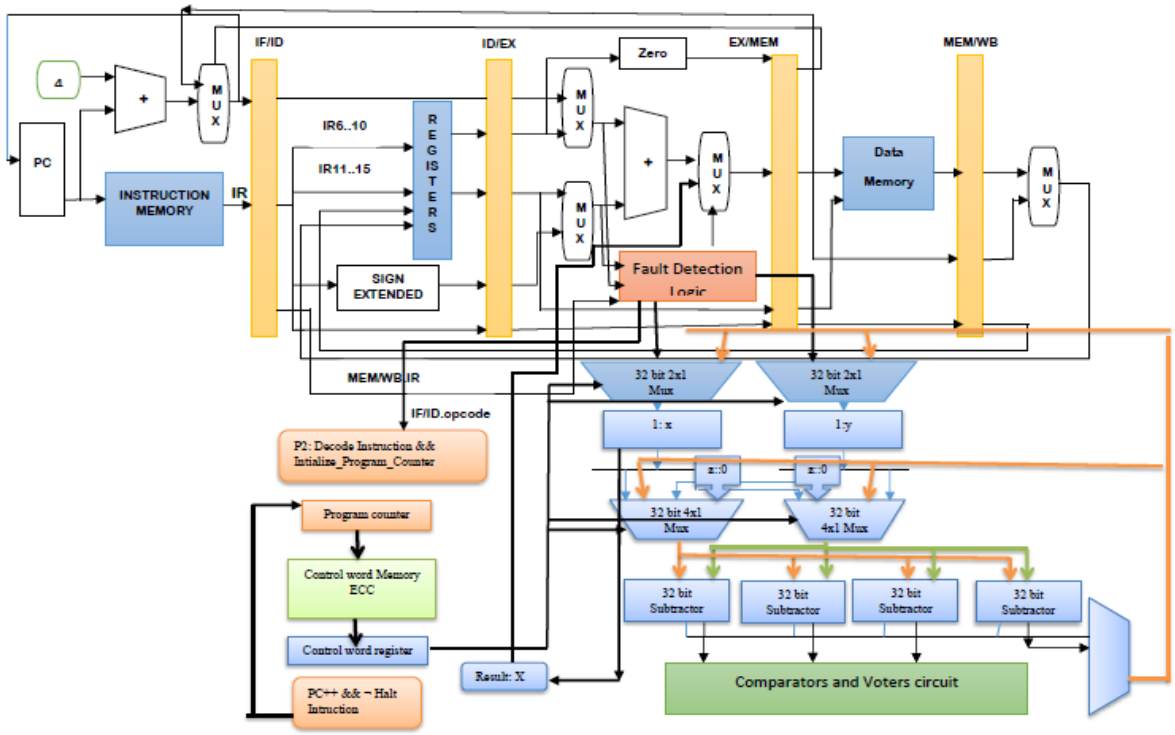


图 1.2 由一个 32 位 MIPS 核心和一个 32 位 OIC 组成的多核系统的微体系结构。

在本节中， MCS-OIC的微体系结构（由一个MIPS 核心和一个 OIC组成）、功耗和面积的仿真结果将予以介绍。 图 1.2 展示了一个多核系统，该系统由一个 32位 MIPS 核心和一个 32 位 OIC组成，在微体系结构中分别占据上半部分和下半部分。MIPS 核心是一个五级流水线标量处理器。取指（IF）、指令译码（ID）、执行（EXE）、存储器访问（MEM）和写回（WB）是 MIPS 流水线的五个阶段。IF/ID、ID/EXE、EXE/MEM 和 MEM/WB 是流水线寄存器。PC 是程序计数器，LMD、Imm、A、B、IR、NPC、Aluoutput 和Cond是临时寄存器，用于在一条指令的时钟周期之间保存状态值。故障检测逻辑（FDL）通过并发执行指令，检测所有算术指令（逻辑指令除外）中的故障。通过比较 ID/EXE.Aluoutput和 FDL 的结果来检测故障。如果发现故障，则暂停流水线。IF/ID.opcode（位于 IR中）以及操作数ID/EXE.A 和 ID/EXE.B被传输至 OIC。IF/ID.opcode被译码，同时ID/EXE.A 和 ID/EXE.B 的值被载入OIC寄存器（X和Y）。OIC.PC被初始化，同时从存储器中将第一个控制字载入其控制字寄存器。在每个时钟周期，控制字寄存器中的控制位被发送至多路复用器的选择线，这些选择线控制减法器的输入线。在每个时钟周期执行减法，以模拟从 MIPS 核心发送的指令。最后，计算结果被载入MEM/WB.Aluoutput，并恢复 MIPS 流水线操作。

图 1.2 所示多核系统（一个 MIPS 核心与一个 OIC）的微架构面积和功耗采用 ASIC 仿真进行估算。采用 FPGA 综合来验证原型（MCS-OIC）的功能，并确定电路中的逻辑元件数量。仿真设置和估算参数的详细信息列于表 1 中。该多核系统总面积为 306283μm2，总功耗为 1.1554W。FDL 占用 6203 μm2 的面积，占系统总面积的 2%。OIC 占用 8122 μm2 的面积，占系统总面积的 2.6%。FDL 的功耗为 1.2mW，OIC 的功耗为 1.4mW，与总功耗相比可忽略不计。对于多核系统的各种配置，系统功耗、系统面积、额外功耗（由 FDL 和 OIC 引起）以及额外面积（由 FDL 和 OIC 引起）列于表 2 中。

表 3 列出了每个 OIC 和每个 MIPS 核心的逻辑单元数量。每个 OIC 的逻辑元件数量仅为一个 MIPS 核心的 2.65%，以及 MCS-OIC 的 2.58%。

表 1 仿真设置和参数评估 - 使用情况

平台	实验设置	参数评估 - 使用情况
FPGA	采用 Cyclone IVE 器件的 Quartus Prime EP4CE115FE29C7	(1) 对逻辑单元数量和寄存器数量进行评估，以比较 OIC 和 MIPS 核心。用法：用于可靠性分析的逻辑元件数量见第 VI 节
ASIC	Cadence Encounter (R) RTL 编译器 RC14.28 –V14.20（Cadence 设计系统 2004）TSMC 90纳米工艺库（tcbn90lphptc 150）。	(1) MCS-OIC 的面积和功耗

表 2 使用 ASIC 仿真对不同配置的多核系统进行功耗和面积估计

配置	系统 功耗(W)	系统 面积（μm²）	额外功耗		额外面积	
			(mW)	% of 系统 功率	(μm2)	% of 系统 area
一个 MIPS 核心 + 一个 OIC	1.1554	306283	2.6	0.2	14325	4.6
一个 MIPS 核心 + 两个 OIC	1.1568	314405	4	0.34	22447	7.1
一个 MIPS 核心 + 四个 OIC	1.1586	338771	5.4	0.46	38691	11.4
两个 MIPS 核心 + 一个 OIC	2.3098	604444	3.8	0.16	20528	3.3
两个 MIPS 核心 + 两个 OIC	2.3112	612566	5.2	0.22	28650	4.6
两个 MIPS 核心 + 四个 OIC	2.3140	628810	8	0.34	44894	7.13
四个 MIPS 核心 + 一个 OIC	4.6222	1175954	6.2	0.13	32934	2.8
四个 MIPS 核心 + 两个 OIC	4.6236	1184076	7.6	0.16	41056	3.46
四个 MIPS 核心 + 四个 OIC	4.6264	1200320	10.4	0.22	57300	4.77
八个 MIPS 核心 + 一个 OIC	9.2430	2393410	11	0.11	57746	2.41
八个 MIPS 核心 + 两个 OIC	9.2444	2401532	12.4	0.13	65868	2.74
八个 MIPS 核心 + 四个 OIC	9.2472	2417776	15.2	0.16	82112	3.39
八个 MIPS 核心 + 六个 OIC	9.25	2434020	18	0.19	98356	4.04

表 3 MCS-OIC 中的逻辑元件

A. 一个 OIC 中的逻辑元件数量	530
B. 一个 MIPS 中的逻辑元件数量 core	19988
C. MCS-OIC 中的逻辑元件数量 （一个包含一个 OIC 的 MIPS 核心）= A + B	20518

基于冗余的核级故障缓解技术/方法，例如[40] LaFrieda等人提出的 Slipstream [39],动态核耦合 (DCC) 方法、可配置隔离[41], Reunion，是[42] Smolens等人提出的指纹识别技术，具有近乎 100% 的面积开销和明显更大的功耗开销。 从表2 可以看出，最大功耗开销和最大面积开销分别为 0.46% 和 11.4%。由表 2 可知，MCS-OIC（一个MIPS核心+ 一个OIC）消耗的面积开销和功耗开销最小，分别为 0.11% 和 2.41%。必须将每条指令的冗余执行降至最低，但仍应提供容错能力。MCS –OIC 仅保护算术指令，而算术指令约占应用程序中全部指令的 60% [4]。

最后，图 1.2 中的 MCS-OIV 架构构成了后续各节所讨论的可靠性模型的基础。 该模型考虑 L 个传统核心（MIPS 核心）和 M 个 OIC（L<=M），以估算系统可靠性。该模型估算不同配置下的系统可靠性，配置范围从 <1 MIPS 核心、1 个 OIC> 紧密耦合，到 <L 个 MIPS 核心、M 个 OIC> 松散耦合。采用 OSS 来描述核心计算方面的可靠性。下一节将概述 OSS模型、OSS 模型参数、MCS-OIC 支持的不同配置，以及 OSS 参数到 MCS-OIC 的映射。

5. MCS-OIC 的 OSS 和模型参数

在 OSS 中，系统预期能够同时执行一个或多个任务。任务可以重复执行，但其执行独立于先前任务的执行。此处介绍的 OSS模型适用于多功能组件（能够执行多种功能的组件称为多功能组件）。MIPS 核心和 OIC 都是多功能组件。OSS 的模型参数包括：(a) 唤醒概率，(b) 启动策略，以及 (c) 组件的准备度。下面将简要说明这些参数。

唤醒概　率：指成功在组件上启动功能的概率。加法　　al
功耗会被　唤醒操作消耗。

启动策略：在系统部署之前，会在组件上测试并激活各项功能，并在其他组件上禁用相同的功能。对于在某个组件上采用启动策略的功能，其唤醒概率为 1。这一点对 OIC 等低功耗设备尤为重要。经过启动策略处理的功能无需唤醒，因此可以节省功耗。

准备度：指在涉及指令迁移的操作发生之前，组件处于就绪状态的概率。在所提出的模型中，组件的初始准备度设为 0.99。后续调用组件时发生故障会导致准备度概率降低。当请求组件提供功能支持时，会采用其之前的准备度来评估系统可靠性。

下一节将讨论采用 OSS 模型参数的 MCS-OIC 的不同配置。

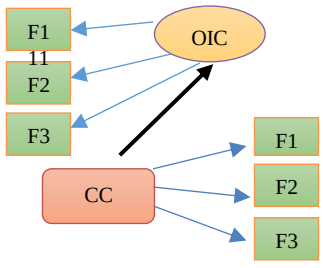
5.1 MCS-OIC 的四种不同配置

通常，考虑一个由 L 个传统核心、M 个 OIC 和 N 个功能组成的系统。针对上述 OSS 的参数影响，可以有三种不同的配置。第一种配置采用传统核心与 OIC 之间的 1:1 映射。在此配置中，每个传统核心均分配一个 OIC，以提供功能支持。第二种和第三种配置采用一对多映射。在第二种配置中，一个传统核心访问多个 OIC。在第三种配置中，多个传统核心同时访问多个 OIC。然而，一个 OIC 不能由多个内核共享。下面将结合图 1.3 中给出的示例对这些配置进行说明。

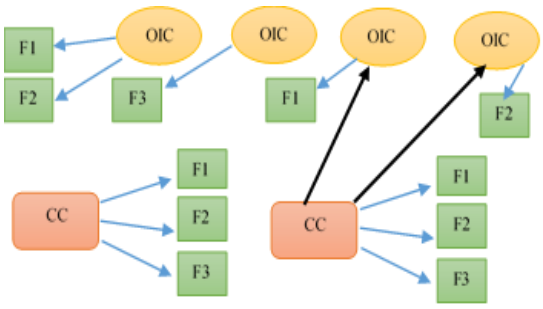
图 1.3 (a) 所示的配置是一个由一个传统核心和一个能够执行三项功能的 OIC 组成的系统。三项功能 F1、F2 和 F3 均正确执行，是系统保持稳定的必要条件。假设三项功能中有一项采用启动策略。其余两项功能可在传统核心以唤醒概率进行唤醒后，在 OIC 上调用。图 1.3 (b) 所示的配置是一个由两个传统核心和四个 OIC 组成的系统。功能 F1、F2 和 F3 均正确执行，是系统保持稳定的必要条件。四个功能配置为四个 OIC，分别为（F1、F2）、F3、F1 和 F2。当两个传统核心中的一个核心发生功能 F1 故障时，将选择准备度高且采用 F1 启动策略的 OIC 来执行任务。若不存在此类 OIC，则选择另一个采用 F1 启动策略的 OIC，或选择具有唤醒概率的 OIC 来执行功能 F1。考虑上述影响因素可以提高系统可靠性，因此图 1.3 (b) 所示配置中的传统核心也采用相同方法来调用功能 F3 和 F2。

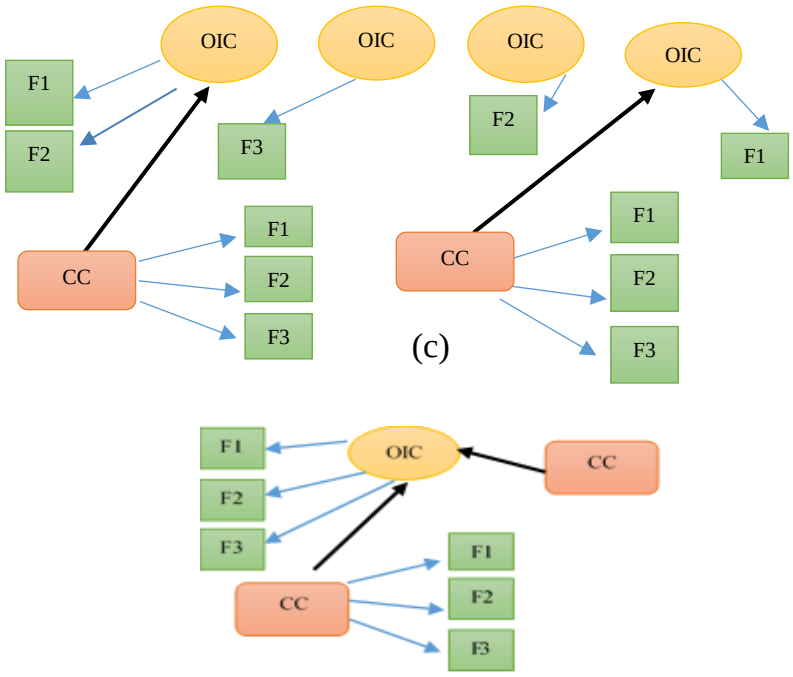
图 1.3 (c) 中的配置是一个由两个传统核心和四个 OIC 组成的系统。系统要保持稳定，必须有三个功能 F1、F2 和 F3 正常运行。四个 OIC 分别配置了功能 (F1, F2)、F3、F2 和 F1。两个不同传统核心中的 F2 和 F1 同时发生故障。系统会选择 OIC 为这两个传统核心提供支持。优先选择已为 F1 和 F2 配置启动策略的 OIC；若此类 OIC 不可用，则调用唤醒 F1 和 F2 概率较高的 OIC。图 1.3 (d) 中的配置是一个由两个传统核心和一个 OIC 组成的系统。在我们的建模中，不考虑两个传统核心同时访问 OIC 的情况。

(a)



(b)





(d)

图 1.3 (a)支持一个传统核心的系统 (b) 由两个 OIC支持的传统核心，(c) 两个传统核心同时访问两个 OIC (d) 同时访问 OIC

5.2 MCS-OIC 的 OSS 模型参数

考虑一个系统，其中包含 L 个活跃传统核心和 M 个候选 OIC。若所有 N 个功能均正常运行，无论是否使用 OIC，都认为系统运行正常。在系统投入使用之前，会对 OIC 上的功能进行测试并激活，每个 OIC 至少激活一个功能。此活动称为启动策略，用变量 x_{ij} 表示。若将 x_{ij} 设为 1，则该功能已准备好执行操作。将 x_{ij} 设为 0，表示功能 j 尚未配置启动策略。

在适当的时候，传统核心会从 OIC 寻求功能支持。在为此任务选择 OIC 之前，会检查 OIC i 是否具备功能 j（记为 a_{ij}）。选择 w 个候选 OIC 来支持传统核心，其中 k 个可能无法响应。Ψ_0 表示执行任务的 OIC 集合。$\Psi_{k,\ b}$ 是组合（$\binom{w}{k}$）的索引。所选 OIC 中，组件级就绪度为 rd_i 且功能级就绪度为 r_{ij} 的 OIC 会被指派执行任务。有时，尽管 OIC 的准备度较高 rd_i，但由于功能 j 级别的就绪度不足（r_{ij}），仍可能无法执行任务。r_{ij} 表示在 OIC 或组件 i 上成功执行功能 j 的概率。具有相似功能且组件级就绪度达标的 OIC 会以唤醒概率 p_{ij}启动。整个操作的可靠性——包括指令迁移以及在 OIC 上执行指令（r_{ij}）——记为 r(t)。系统的可靠性等于 OIC 的就绪度与 OIC 上所执行功能的可靠性的乘积。上述参数按讨论顺序汇总于表 4。

当前版本是版本 1.0。 稿件正在审稿中。修改版或扩展版将刊登于期刊。

表 4 模型参数和符号说明

符号说明	描述
L	系统中独立运行的传统核心数量。
M	候选OIC 或组件的集合 $\{1,2,3 \ldots m\}$,L ≤ M
N	功能集合 $\{1,2,3 \ldots n\}$
x_{ij}	如果在启动策略中分配 i^{th} OIC 执行功能 j，则为 1；否则为 0。
A	包含决策变量的矩阵。$i \in M, j \in N$。
a_{ij}	如果功能 j 在 i^{th} OIC 上可用，则为 1；否则为 0。
w	用于构建系统的候选 OIC 数量，$w \leq m$。
Ψ_0	所有已选 OIC 的集合 $\lvert\Psi_0\rvert = w$
$\Psi_{k,b}$	b^{th} 其中有 k 个 OIC 尚未就绪的 Ψ_0 非空子集。$0 \leq k \leq w-1, 1 \leq b \leq \binom{w}{k}$ 和 $\Psi_{0,\ 1} = \Psi_0$。
rd_i	OIC i 的准备度等级。
p_{ij}	唤醒概率：在 i^{th} OIC 上启用功能 j 的概率，$i \in M, j \in N$
$r(t)$	请求执行故障的活动核心的可靠性指令。它包含 OIC i 上功能 j 的可靠性 $r_{ij}{}^{th}$。
r_{ij}	OIC i 上功能级准备度 j^{th}。它是 r(t) 的组成部分。
S^R	系统可靠性
C_{ij}	成本 C_{ij} 是 OIC i 上功能 j 的执行时间（以时钟周期计）。
C	传统核心从一组 OIC 中寻求计算时的最大等待时间。
U	$<u_1, u_2, \ldots, u_N>$, u_i 表示功能 j 的副本数量，位于系统。它是 A 中每一列的总和。
X_{mxn}	包含决策变量的矩阵。$i \in M, j \in N$。
I_j	能够执行功能 j 的候选 OIC 子集。
sgn (z)	当 z > 0 时为 1；否则为 0。

下一节将结合仿真结果评估 MCS-OIC 系统可靠性的理论可靠性模型，并介绍 MCS-OIC 的四种特殊情况。

6. MCS-OIC的系统可靠性

6.1 可靠性建模假设

(a)所有 OIC 在组件层面（或其核心层面）相互独立；同样，对于每个 OIC，rd_i 和 p_{ij}相互独立。

(b)当且仅当所有 n 个功能均正常工作时，才称该系统正常运行。

当前版本为版本 1.0。 稿件正在审稿中。修改版或扩展版将在期刊上发表。

(c)当 OIC 未准备就绪时，OIC 的所有功能都会停止。

(d)所有 OIC 均为多功能型。

(e)所有 OIC 均为热备用冗余核心。OIC 或组件级就绪度并非瞬时完成。

6.2. 系统可靠性的推导

考虑启动策略$X_{m\,x\,n}=[x_{ij}]$，以及功能 j 在i^{th} OIC 上的可用性$A_{m\,x\,n}=[a_{ij}]$，两种条件下的系统可靠性，即k= 0 和k > 0，分别由方程（14）和（15）给出。变量 k 表示发生故障的 OIC 数量。

$$S^R = \sum_{k=0}^{w-1}\sum_{b=1}^{\binom{w}{k}}\{[\prod_{u\in\Psi_{k,b}} rd_u].[\prod_{v\in\Psi_0/\Psi_{k,b}}(1-rd_v)].\prod_{j=1}^{n}[1-\prod_{i\in\Psi_{k,b}}(1-r(t)\ .E_{ij})^{a_{ij}}]\}, \text{when } k > 0. \quad (14)$$

$S^R = [\prod_{u\in\Psi_0} rd_u].\prod_{j\in N}[1-\prod_{i\in\Psi_0}(1-r\ t\ E_{ij})^{a_{ij}}]$（()），当 k = 0。 (15)

推导如下。

当所有功能都由传统核心执行时，认为系统正在运行。当传统核心中的少数功能无法正常工作时，会选择 OIC 来可靠地执行这些功能。“组件”和 OIC 可互换使用。对于给定选定的 $\Psi_{k,\,b}$ OIC，所有功能均得到执行的概率由方程 (16) 给出。

P {所有功能均正常运行 | $\Psi_{k,\,b}$ }

$= \prod(1 - P\{\ all\ selected\ OICs\ fail\ to\ perform\ function\ j\ |\Psi_{k,b}\}_{j=1}^{n})$

$= \prod_{j=1}^{n}[1-\prod_{i\in\Psi_{k,b}} \Pr\{\ ith\ OIC\ fails\ to\ perform\ function\ j)\}]$ (16)

为了估计 i^{th} OIC 无法执行功能 j 的故障，引入了 E_{ij} a 变量，该变量考虑了启动策略和唤醒概率，其计算公式如方程（17）所示。

$$E_{ij} = \begin{cases} p_{ij}\ if\ x_{ij} = 0 \\ 1\ if\ x_{ij} = 1 \end{cases} \quad (17)$$

现在，E_{ij} 可以用 x_{ij} 表示，如方程 (18) 所示。

$$E_{ij} = (1 - p_{ij})x_{ij} + p_{ij} \quad (18)$$

现在，i^{th} OIC 使用 E_{ij} 未能执行功能 j 的概率表示在方程（19).

$$P\ (i^{th}\ \text{OIC}\ fails\ to\ perform\ function\ j) = (1 - r(t)E_{ij})^{a_{ij}} \quad (19)$$

考虑在所选项中以 $\Psi_{k,b}$ 表示的优良 OICs，给定 $\Psi_{k,b}$ 时所有功能正常工作的概率由方程 (20) 给出。

$$P(all\ functions\ are\ performing|\Psi_{k,b}) = \prod \left[1 - \prod_{j=1}^{n} \quad \prod_{i\in\Psi_{k\,b}} \quad 1 - r\ t\ E_{ij}^{\ a_{ij}}\right]0), \tag{20}$$

将方程 (19) 代入方程 (16)，即可得到方程 (20)。在方程 (20) 中，r(t) 包含 r_{ij}，它表示组件 i 上功能 j 的可靠性。下面讨论系统中 OIC 的准备度。选取 w 个 OIC，其中 k 个发生故障。k 的取值范围为 $k = 0 \ldots w\text{-}1$。共有 $\binom{w}{k}$ 或 wC_k 种可能的选项。令 b 表示这些选项的序号。（$b = 1 \ldots \binom{w}{k}$）。令 $\Psi_{k,b}$ 表示 Ψ_0 的第 b 个非空子集，其中 k 个 OIC 未就绪。k 和 b 的取值范围为：$0 \le k \le w-1$，$1 \le b \le \binom{w}{k}$。可用组合的准备度由方程 (21) 给出。

$$Readiness\ Probability = \left[\prod_{u\in\Psi_{k,b}} rd_u\right].\left[\prod_{v\in\Psi_0/\Psi_{k,b}}(1 - rd_v)\right] \tag{21}$$

rd_u 和 rd_v 是 OIC 的准备度。因此，系统可靠性由方程（22）给出。

$$S^R = \sum_{k=0}^{w-1}\sum_{b=1}^{\binom{w}{k}}\left\{\left[\prod_{u\in\Psi_{k,b}} rd_u\right].\left[\prod_{v\in\Psi_0/\Psi_{k,b}} 1 - rd_v\right]\right\} .\{\text{probability of all functions working with given } \Psi_{k,b}\} \tag{22}$$

将方程 (20) 代入方程 (22)，即可得到上文所示的方程 (14)。

让我们考虑两个示例，以说明通过增加启动策略功能以及功能或组件级冗余，可以提高方程 (15) 中的系统可靠性。示例 1：包含三个 OIC 的系统，每个 OIC 有三个功能；示例 2：包含五个 OIC 的系统，每个 OIC 有五个功能。对于这两个示例，假设准备度 $rd_i = 0.99998$，且 r(t) = 0.65。图 1.4 显示，随着纳入启动策略的功能数量增加，系统可靠性也随之提高。如图 1.5 所示，OIC 数量、每个 OIC 中的功能数量以及纳入启动策略的功能数量增加，都会提高系统可靠性。三个 OIC 且每个 OIC 有三个功能时，系统可靠性最高可达 0.9999012；五个 OIC 且每个 OIC 有五个功能时，最高可达 0.99998。图中 X轴表示纳入启动策略的功能数量，Y轴表示可靠性。

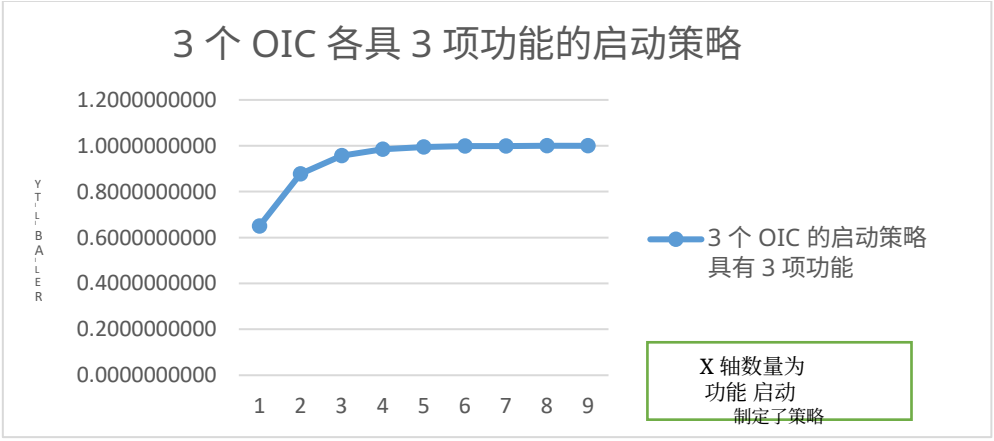


图 1.4　　可靠性与 3 个 OIC 的启动策略功能数量之间的关系，每个 OIC 均具有三个功能　　s.

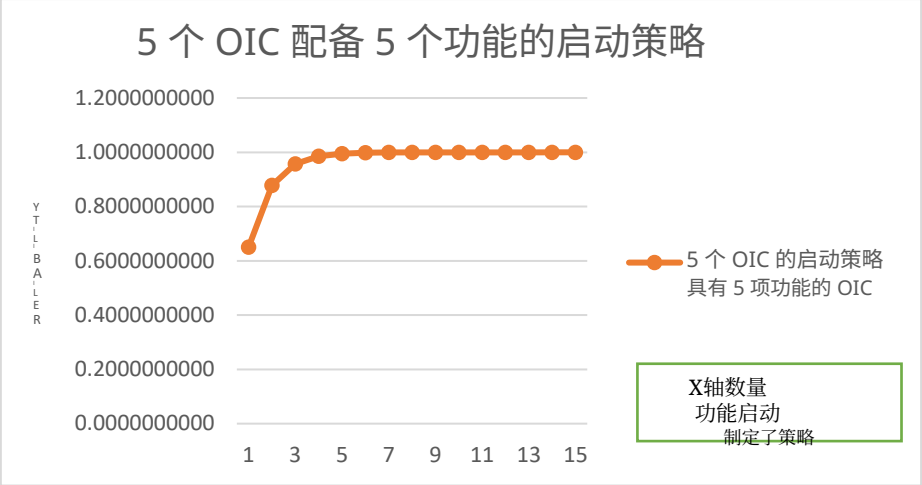


图 1.5 5 个 OIC 中每个包含五项功能时，可靠性与启动策略化功能数量的关系。

方程 (15) 有四种特殊情况，分别是：(a) OIC的准备度相同(b)功能数量为 (N = M) 的系统等于具有 $a_{ii} = p_{ii} = 1$ 的 OIC 数量(c) 所有功能在所有OIC上均可用，且p_{ij} 等于 1，以及(d)具有相同组件的系统 ($r_{ij} = R_j$ 和 $p_{ij} = P_j, j = 1, 2, 3,..., n$)。

下面将考察四种特殊情况。

特殊情况 1：

如果所有 OIC 的准备度都相同，即 $rd_i = rd, i = 1, 2, 3...m$，则当 $k > 0$ 时，系统的可靠性由方程 (23) 给出。

$$S^R = \sum_{k=0}^{w-1} \sum_{b=1}^{\binom{w}{k}} \{[rd^{w-k}].[(1-rd)^k].\prod_{j=1}^{n}[1 - \prod_{i\in\Psi_{k,b}}(1 - r(t).E_{ij})^{a_{ij}}]\} \quad (23)$$

在方程 (23) 中，通过假设所有正常 OIC 的准备度相同来估算系统可靠性。因此，系统可靠性是所有正常 OIC 的准备度与正在运行的传统核心的可靠性 r(t) 的乘积，其中包括 r_{ij} （OIC i 上功能 j 的可靠性）。

当 $k = 0$ 时，系统可靠性由方程 (24) 给出。

$$S^R = [rd^w].\prod_{j=1}^{n}[1 - \prod_{i\in\Psi_{k,b}}(1 - r(t).E_{ij})^{a_{ij}}] \quad (24)$$

方程 (24) 在假设所有 OIC 均正常工作且准备度相同的情况下计算系统可靠性。考虑到 $k = 0$ and $rd_i = rd$，方程 (24) 中的系统可靠性是所有 OIC 的准备度与正在运行的传统核心的可靠性 r(t) 的乘积，其中包括 r_{ij} （OIC i 上功能 j 的可靠性）。

特殊情况 2：

在假设系统正常运行，即所有功能都由至少一个组件或核心执行的前提下，考虑这样一种特殊情况：功能数量与组件数量相等（m = n）。功能 j 分配给 i^{th} OIC，矩阵 X 将是一个单位矩阵。即（m = n），$\forall\, i, j \in N(i \neq j)$，$a_{ii} = p_{ii} = 1, a_{ij} = p_{ij} = 0$。启动策略矩阵 X 是一个单位矩阵，系统的可靠性推导如下。

特殊情况 2 的推导：

由上述陈述可清楚看出，X矩阵是单位矩阵。由方程 5.5可知$E_{ij} =(1 - p_{ij})x_{ij} + p_{ij}$。显然，下面的方程 (25) 成立。

$$p_{nxn} = E_{nxn} = X_{nxn} = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix}, \Psi_0 = N, w = n \tag{25}$$

由于 $a_{ij} = 0$ 和 $\forall k \neq 0, \forall i \in \Psi_{k,b}, i \neq j$，X 矩阵中由非对角元素表示的功能的可靠性为零，将方程 (25) 代入方程 (14) 可得方程 (26)。

$$\prod_{j\in N}\left[1 - \prod_{i\in\Psi_{k,b}}(1 - r(t)E_{ij})^{a_{ij}}\right] = 0 \tag{26}$$

在方程 (26)中，当i ≠ j 时，组件 i 上函数 j的可靠性为零。对于每个 i、j，j ∈ N($i \neq j$)，$\Psi_0 = N, a_{ii} = 1$，$a_{ij} = 0$, $we\ have\ (1 - r(t)E_{ij})^{a_{ij}} = 1$。

$$For\ i = j, we\ have\ (1 - r(t)E_{ij})^{a_{ij}} = (1 - r(t).1)^1.$$

因此，方程 (26) 推导如下：

$$\prod_{j\in N}\left[1 - \prod_{i\in \Psi_0}(1 - r(t)q_{ij})^{a_{ij}}\right] = \prod_{j\in N}[1 - (1 - r(t).1)^1]] = \prod_{j\in N} r(t) \tag{27}$$

将方程 (27) 代入方程 (15)，得到方程 (28)。

$$S^R = \prod_{j\in N} rd_j r(t) \tag{28}$$

在这些条件下，系统构成串联系统。

特殊情况 3：

考虑到 $m = n$、$\forall i$、$j \in N$、$a_{ij} = p_{ij} = 1$，系统的可靠性推导如下：

特殊情况 3 的推导：

$$p_{nxn} = E_{nxn} = \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix}, \Psi_0 = N, w = n \tag{29}$$

在方程 (29) 中，考虑以下假设：功能数量为 = OIC 数量，并且所有功能在所有 OIC 上均可用，唤醒概率为 $p_{ij} = 1$，E_{ij} 应为全 1 矩阵。

将方程 (29) 代入方程 (14)，得：

$$S^R = \sum_{k=0}^{w-1} \sum_{b=1}^{\binom{w}{k}} \{[\prod_{u \in \Psi_{k,b}} rd_u].[\prod_{v \in \Psi_0/\Psi_{k,b}} (1 - rd_v)].1\} \quad (30)$$

在方程 (30) 中，当所有 OIC 上的所有功能均可用时，估算系统可靠性。因此，由于所有 OIC i 上的所有功能 j 的可靠性均为 1，方程 (30) 可简化为所有正常运行的 OIC 的准备度之积。

已知P{所有组件均失效 } + P {至少有一个组件存活 }= 1。

$$\prod_{v \in N} (1 - rd_v) + \sum_{k=0}^{w-1} \sum_{b=1}^{\binom{w}{k}} \{[\prod_{u \in \Psi_{k,b}} rd_u].[\prod_{v \in \Psi_0/\Psi_{k,b}} (1 - rd_v)]\} = 1$$

$$\prod_{v \in N} (1 - rd_v) + S^R = 1$$

$$S^R = 1 - \prod_{v \in N} (1 - rd_v) \quad (31)$$

在这些条件下，该系统类似于并联系统。

特殊情况 4:

对于由相同 OIC 组成的系统，即 $\forall i$、 $p_{ij} = P_j$、 $and\ X_j = \{ i | x_{ij} = 1\}$，对于情况 k > 0 ，系统的可靠性由方程 (32) 给出；对于情况 k = 0，系统的可靠性由方程 (33) 给出。

$$S^R = \sum_{k=0}^{w-1} \sum_{b=1}^{\binom{w}{k}} \left\{[\prod_{u \in \Psi_{k,b}} rd_u].[\prod_{v \in \Psi_0/\Psi_{k,b}} (1 - rd_v)].\prod_{j=1}^{n} \left[1 - (\prod_{i \in \Psi_{k,b} \setminus X_j} \left(1 - r(t) P_j\right)^{a_{ij}} .\prod_{\in \Psi_{k,b} \cap X_j} (1 - r(t))^{a_{ij}}\right]\right\} \text{ when k } > 0 \quad (32)$$

$$S^R = [\prod_{u \in \Psi_0} rd_u]。\prod_{j=1}^{n} \left[1 - \prod_{i \in \Psi_{k\,b}\ X_j} 1 - r\ t \quad P_j^{\ a_{ij}} \prod_{\in \Psi_{k\,b} \cap X_j} 1 - r\ t \quad ^{a_{ij}}\right]((()), \setminus。 (())), \quad 当\ \mathrm{k} = 0 \quad (33)$$

根据这一前提，S^R 可写为:

$$S^R = \sum_{k=0}^{w-1} \sum_{b=1}^{\binom{w}{k}} \{[\prod_{u \in \Psi_{k,b}} rd_u].[\prod_{v \in \Psi_0/\Psi_{k,b}} 1 - rd_v].\prod_{j=1}^{n} [1 - \prod_{i \in \Psi_{k,b}} (1 - r(t).E_{ij})^{a_{ij}}]\} \quad (34)$$

其中 $E_{ij} = p_j\ if\ x_{ij} = 0$、 i。 e。 ， $i \notin X_j$、 $otherwise$ 1。

$\prod (1 - r(t)^{\ i \in \Psi_{\mathrm{h},\ b}}$。 $E_{ij})^{a_{ij}}$ 可拆分为

$$(\prod_{i \in \Psi_{k,b} \setminus X_j} \left(1 - r(t) P_j\right)^{a_{ij}} .\prod_{\in \Psi_{k,b} \cap X_j} (1 - r(t))^{a_{ij}}) \quad (35)$$

在这些 e条件下，系统可靠性是具有和不具有唤醒概率的 OIC 的可靠性乘积。

当前版本为版本 1.0。 稿件正在审稿。修改或扩展后的版本将在期刊上发表。

为对寻求 OIC功能支持的活动传统核心的失效率进行建模，使用Erlang分布。在 Erlang 分布中，[43]、 β（形状参数）为整数，可靠性函数的闭式表达式见方程（36）。该模型包含一个活动组件和 β -1 个相同组件。

$$R(t) = \sum_{kl=0}^{\beta-1} \frac{(\lambda t)^{kl} e^{-\lambda t}}{kl!} \tag{36}$$

将方程 (36) 中的 r(t) 代入方程 (14) 和 (15)，当 L = 1 给定时，系统可靠性由方程 (37) 和方程 (38) 给出。

$$S^R = \int_0^{\infty} \sum_{k=0}^{w-1} \sum_{b=1}^{\binom{w}{k}} \left\{ \left[\prod_{u \in \Psi_{k,b}} rd_u \right] . \left[\prod_{v \in \Psi_0 / \Psi_{k,b}} 1 - rd_v \right] . \prod_{j=1}^{n} \left\{ (1 - \prod_{i \in \Psi_{k,b}} (1 - \sum_{\beta=0}^{|\Psi_{k,b}|} \frac{(\lambda t)^{\beta} e^{-\lambda t}}{\beta!} . E_{ij})^{a_{ij}} \right\} \right\} dt \quad \text{when } k > 0.$$

(37)

在方程(37)中，当对选定的 w 个 OIC（发生 k 次故障）估计 L = 1 时的系统可靠性 s.

$$S^R = \int_0^{\infty} \left[\prod_{u \in \Psi_0} rd_u \right] . \prod_{j \in N} \left[1 - \prod_{i \in \Psi_0} \left(1 - \sum_{\beta=0}^{|\Psi_0|} \frac{(\lambda t)^{\beta} e^{-\lambda t}}{\beta!} E_{ij} \right)^{a_{ij}} \right] dt \quad \text{when } k = 0. \tag{38}$$

在方程 (38) 中，当 L = 1 情形下选定的 OIC 中没有发生故障时，对系统可靠性进行估算。L > 1 情形下 Erlang 分布的可靠性函数见方程 (39)。

$$R(t) = \sum_{kl_i=0}^{\beta-1} e^{-\sum_{i=1}^{L} \lambda_i t} . \prod_{i=1}^{L} \frac{(\lambda_i t)^{kl_i}}{kl_i!} \quad , i = 1, \dots, L \tag{39}$$

因此，具有 k> 0 和 k = 0 的 L 个独立运行的传统核心的系统可靠性分别如方程 (40) 和 (41) 所示。

$$S^R = \sum_{k=0}^{w-1} \sum_{b=1}^{\binom{w}{k}} \left\{ \left[\prod_{u \in \Psi_{k,b}} rd_u \right] . \left[\prod_{v \in \Psi_0 / \Psi_{k,b}} 1 - rd_v \right] . \prod_{j=1}^{n} \left[1 - \prod_{i \in \Psi_{k,b}} (1 - \sum_{kl_{ii}=0}^{|\Psi_{k,b}|} e^{-\sum_{ii=1}^{L} \lambda_{ii} t} . \prod_{ii=1}^{L} \frac{(\lambda_{ii} t)^{kl_{ii}}}{kl_{ii}!} . E_{ij})^{a_{ij}} \right] \right\}, ii = 1, \dots, L \quad \text{when } k > 0 \tag{40}$$

$S^R = \left[\prod_{u \in \Psi_0} rd_u \right] \prod_{j \in N} \left[1 - \prod_{i \in \Psi_0} 1 - \sum_{kl_{ii}=0}^{|\Psi_0|} e^{-\sum_{ii=1}^{L} \lambda_i t} \prod_{i=1}^{L} \frac{\lambda_{ii} t^{kl_{ii}}}{kl_{ii}} E_{ij}^{a_{ij}} \right] ii = 1 \quad L \quad = . (. 0!) , , \dots ,$ 当 k 0

(41)

方程 (41) 的特殊情况是由一个传统核心和一个 OIC 组成的系统，用于执行所有 j 项功能。其可靠性由方程 (42) 给出。

$$S^R = rd_{i=1} . \prod_{j=1}^{n} \left[1 - (1 - r(t) E_{1j})^{a_{ij}} \right] \tag{42}$$

现在，我们考察一个包含多个传统核心和一个 OIC 的系统的系统可靠性，具体如下所示。

采用方程（40）中所述的系统方程，图 1.6 和图 1.7 绘出了分别由两个独立传统核心（L=2）和（M=2、3、4）个 OIC，以及三个独立传统核心（L=3）和（M=3、4、5）个 OIC 组成的系统的可靠性。方程中提到的可靠性函数参数 λ

(39) 是电路中逻辑单元数量的函数。每个 OIC、每个 MIPS 核心以及每个MCS-OIC（包含一个 OIC 和一个MIPS 核心）的逻辑单元数量，均通过 FPGA 综合确定，并列于表3 中。微架构组件中的逻辑单元数量越多，其可靠性就越高。显然，OIC 数量的增加将提高系统可靠性，这一点已在图1.6和图 1.7 中观察到。也就是说，通过上述观察结果证实了使用低功耗 OIC 的多核系统可靠性得到了提升。

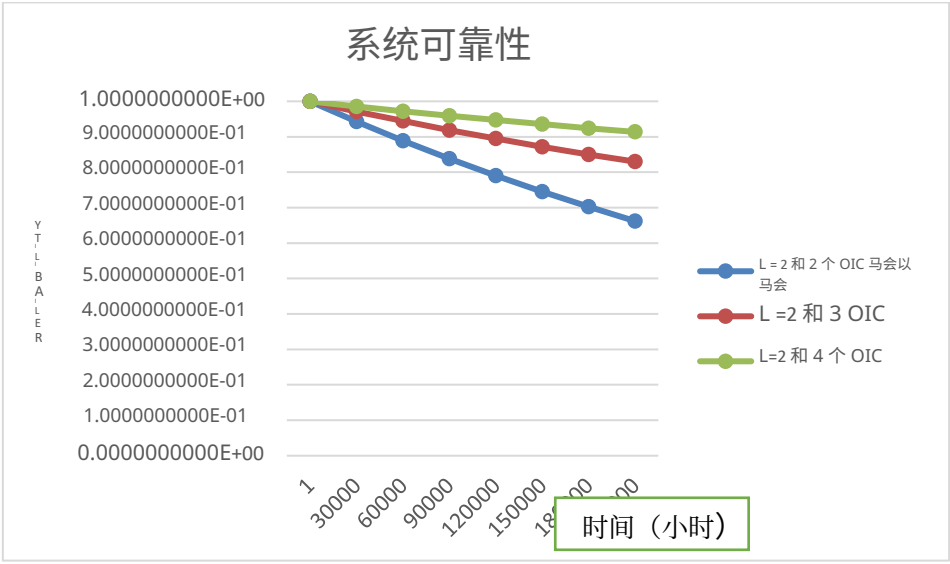


图 1.6 含有 2 个独立传统核心和 2、3、4 个 OIC 的多核系统的系统可靠性与时间（小时）的关系。

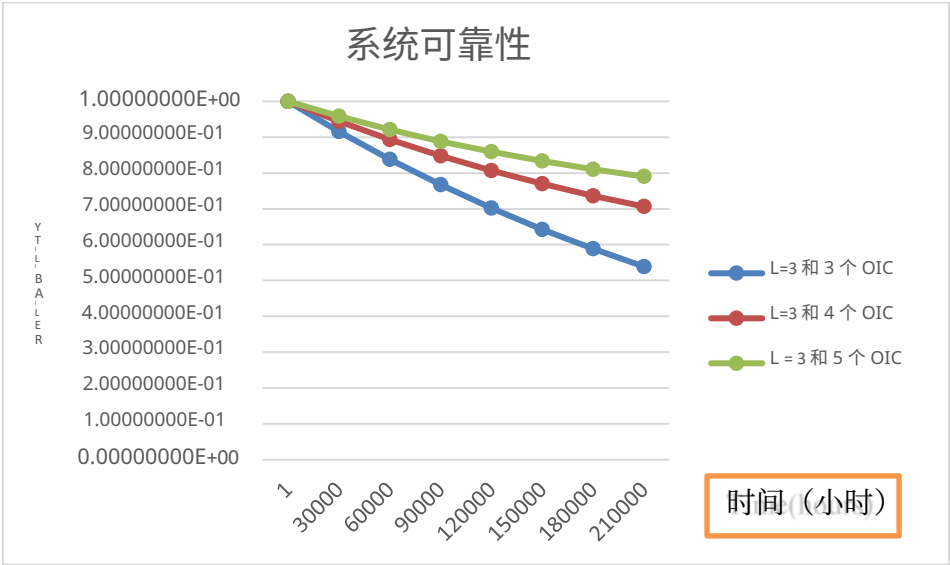


图1.7 由 3 个独立传统核心和 3、4、5 个 OIC 组成的多核系统的系统可靠性与时间（小时）的关系。

7. 区间 RAP 及其固有特性

如第五节所述，在 MCS-OIC 中，OIC 的功能映射到传统内核有三种配置。第一种配置为 1:1 映射，即将 OIC 中的功能直接映射到传统内核。第二种和第三种配置涉及一对多映射，此时，选择传统内核所需的 OIC 及其功能，成为决定系统可靠性的重要考量因素。对 OIC 及其功能的这种选择被视为冗余分配问题。本节将介绍 RAP 的建模方式及其固有特性。

如上一节所述，在基于 OSS 的模型中，随着 OIC 中启动策略所涵盖的功能数量 (X) 增加，以及功能级或组件级冗余 (A) 增加，系统可靠性也会提高。然而，在 OIC 上执行这些功能会产生性能开销，因此需要付出相应成本。因此，必须在开销成本 C 的约束下，通过确定 X 和 A 的取值来优化系统可靠性。此外，还需考虑另一因素：将指令从传统内核迁移到 OIC 并在 OIC 上执行功能的可靠性 r(t) 并不精确。这种不确定性可能源于制造复杂性、设计复杂性或部署环境。区间分析可用于处理这种不确定性。因此，为 r(t) 指定区间值。由此，RAP 被称为区间 RAP。

7.1. 区间 RAP 的形式化

方程 (15) 在下方的方程 (43) 中重新表述，该方程的目标函数是在所选 OIC 中没有故障的情况下，通过决策变量 X 和 A 最大化系统可靠性。

最大化

$$S^R(X,A) = \left[\prod_{i\in\Psi_0} rd_i\right].\prod_{j\in N}\left[1-\prod_{i\in\Psi_0}(1-r(t)E_{i,j})^{a_{i,j}}\right] rd_i\,, r(t)\in[0,1], 0 < t_l < t < t_r. \text{when L} \le \text{M and k} = 0 \quad (43)$$

受以下约束条件限制：

$$E_{ij} = (1-p_{ij})x_{ij} + p_{ij}, p_{ij} \in [0,1] \quad (44)$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} C_{ij}\, x_{ij} \le C \quad (45)$$

$$x_{ij} \in [0,1] \quad (46)$$

决策变量 X 表示启动策略 x_{ij}，A 表示可用性 a_{ij}。在非线性约束（公式（45））中，成本 C_{ij} 表示组件 i 上函数 j 的执行时间（以时钟周期计）。C 表示传统核心从给定的一组 OIC 中寻求计算时的最大等待时间。$t \in [t_l, t_r]$ 表示区间时间。定义传统核心从 OIC 寻求功能支持的事件可能在区间时间内变化，称为区间任务时间。在该区间内，当任务时间达到下界时，OIC 的可靠性可能达到上界。当任务时间达到上界时，OIC 的可靠性可能达到下界。这意味着，对于 $t = t_L$，$r(t_L) = r_R$；对于 $t = t_R$，$r(t_R) = r_L$。现在，可根据定义 1.2 重新表述目标函数，如公式（47）所示。公式（47）中的表述是一个带区间系数的非线性整数规划问题。

最大化

$$S^R(X,A) = \begin{bmatrix} \prod_{i\in\Psi_0} rd_i \cdot \prod_{j\in N}\left[1 - \prod_{i\in\Psi_0}(1 - r_R E_{i,j})^{a_{i,j}}\right], \\ \prod_{i\in\Psi_0} rd_i \cdot \prod_{j\in N}\left[1 - \prod_{i\in\Psi_0}(1 - r_L E_{i,j})^{a_{i,j}}\right] \end{bmatrix} \quad (47)$$

约束保持不变。

B. 所提出的 RAP 系统的重要特征

下文将讨论 RAP 的重要固有特征。

(a) 单调性：$S^R(x_{ij},a_{ij})$ 随 a_{ij} $(i = 1... m; j = 1...n)$ 单调递增。

证明

我们知道，$r_{ij} \in [0,1]$、$E_{ij} \in [0, 1]$ 和 $r(t)\in[0,1]$。r_{ij} 是 r(t)和 $a_{ij} \geq 0$ 的组成部分。已知$(1- r(t)E_{ij})^{aij} \in [0,1]$。i* 和 j*分别为 i 和 j 的任意值，且是固定不变的。对于两个实例$a_{i^*j^*(2)}$、$a_{i^*j^*(1)}$，如果 $a_{i^*j^*(2)} > a^{1}_{i^*j^*}()$，则会出现两个条件 S1 和 S2，具体如下所述。

S1：考虑 $a_{i^*j^*(1)} \geq 1$ $and\ that\ \Psi_0(2) = \Psi_0(1)$，$\Psi_0(2)$ 和 $\Psi_0(1)$ 中所选组件的子集相同。考虑 $F \subset \Psi_0(1) = \Psi_0(2)$。对于不包含 i^* 的 F，使用方程 (48) 确定的$a_{ij} = a^{2}_{i^*j^*}()$ 和$a_{ij} = a^{1}_{i^*j^*}()$ 的系统可靠性相同。但是，对于包含i^* 和 $a_{i^*j^*(2)} > a^{1}_{i^*j^*}()$ 的 F，表达式 (49) 中的不等式成立。因此，包含$a_{i^*j^*(2)}$ 的所有子集 F 的总和大于或等于包含$a_{i^*j^*(1)}$ 的 F。

$$S^R = \left[\prod_{u\in F} rd_u\right].\left[\prod_{v\in\Psi_0\setminus F}(1 - rd_v)\right].\prod_{j=1}^{n}\left[1 - \prod_{i\in F}(1 - r(t)E_{ij})^{a_{ij}}\right] \quad (48)$$

$$(1 - r(t)^*E_{i^*j^*})^{a^{(2)}_{i^*j^*}} \leq (1 - r(t)^*E_{i^*j^*})^{a^{(1)}_{i^*j^*}}, where\ r_{i^*j^*} \text{ 是 r(t)* 的组成部分。} \quad (49)$$

S2：考虑 $a_{i^*j^*(1)} = 0$，则 $a_{i^*j^*(2)}$ 大于零，且 $i^* \in \Psi_0(2)$。但是，i^* 不属于 $\Psi_0(1)$。对于 $F^{(2)} \subset \Psi_0(2)$ 和 $F^{(1)} \subset \Psi_0(1)$，对条件 S3 和 S4 进行了分析，讨论如下。

S3：i^* 不属于 $F^{(2)}$ $and\ F^{(1)}$。使用 $F^{(2)}$ $and\ F^{(1)}$ 确定的系统可靠性相同，并由方程 (50) 给出。

$$\left[\prod_{u\in F^{(2)}} rd_u\right].\left[\prod_{v\in\Psi_0\setminus F^{(2)}}(1 - rd_v)\right].\prod_{j=1}^{n}\left[1 - \prod_{i\in F^{(2)}}\left(1 - r(t)E_{ij}\right)^{a_{ij}}\right] =$$

$$\left[\prod_{u\in F\,1} rd_{u\,0}\right].\left[\prod\left(_{v\in\Psi_0\ F\,1}\ 1 - rd_v\right)\setminus()\right].\prod_{j=1}^{n}\left[1 - \prod_{i\in F\,1}\ 1 - r\ t\ E_{ij}\ ^{a_{ij}}\right](())() \quad (50)$$

S4：i* 属于$F^{(2)}$，满足$F^{(2)} = F^{(1)} \cup \{i^*\} \subset \Psi_0(2)$且$F^{(1)} \subset \Psi_0(1)$。使用$F^{(2)}$ 确定的系统可靠性大于或等于使用 F(1)确定的系统可靠性。因此，表达式 (51) 中的以下不等式成立。

$$\left[\prod_{u\in F^{(2)}} rd_u\right].\left[\prod_{v\in\Psi_0\setminus F^{(2)}}(1-rd_v)\right].\prod_{j=1}^{n}\left[1-\prod_{i\in F^{(2)}}\left(1-r(t)E_{ij}\right)^{a_{ij}}\right] \geq$$

$$\left[\prod_{u\in F^{(1)}} rd_u\right].\left[\prod_{v\in\Psi_0\setminus F^{(1)}}(1-rd_v)\right].\prod_{j=1}^{n}\left[1-\prod_{i\in F^{(1)}}\left(1-r(t)E_{ij}\right)^{a_{ij}}\right] \quad (51)$$

从方程 (51) 可以明显看出，使用$F^{(2)}$ 确定的系统可靠性大于或等于由$F^{(1)}$产生的系统可靠性。额外组件 i* 被纳入$F^{(2)}$。系统可靠性的提升得到了证明。

证明说明：

假设两个实例 $a_{i^*j^*(2)}$、$a_{i^*j^*(1)}$ 和 $\Psi_0(1) = \Psi_0(2)$ 成立。在实例 (2) 和 (1) 中，选定组件的数量相同，则使用方程 (48) 计算出的可靠性相同。如果实例(2) 的 F 包含 i*，且 $a_{i^*j^*(2)} > a_{i^*j^*(1)}$，则不等式 (49) 成立。假设 i* 组件不属于两者 $F^{(2)}\ and\ F^{(1)}$，则使用 $F^{(2)}\ and\ F^{(1)}$ 在方程 (50) 中计算出的可靠性相同。假设 $F^{(2)} = F^{(1)} \cup \{i^*\} \subset \Psi_0(2)$ 成立，则使用 $F^{(2)}$ 计算出的系统可靠性大于或等于使用 $F^{(1)}$ 计算出的系统可靠性，如方程 (51) 所示。

6.3 函数 j 的可靠性：

函数 j 对所选组件的可靠性可以进行评估，并由方程 (52) 给出。

$$r_{fj} = 1 - \prod_{i\in I_j}\left[1 - rd_i\,(1-(1-r_{ij}.E_{ij})^{a_{ij}}\right] \quad (52)$$

r_{ij} 是 r(t) 的组成部分。I_j 表示能够执行功能 j 的组件索引集。

6.4 最大系统可靠性：

假设系统的 A(a_{ij}) 已知，为了最大化系统可靠性，最佳启动策略初始化为 1，即 $x_{ij} = 1$ 对于函数 j，其中

$$\tilde{\iota_j} = arg\ \max_{i\in\Psi_0}\left\{1-\prod_{l\in I_j}\left[1-rd_l(1-(1-r_{lj}.E_{lj}^{(i)})^{a_{ij}})\right]\right\}\ E_{lj}^{(i)} = 1, if\ l = i; otherwise\ p_{lj} \quad (53)$$

8. 区间 RAP 的遗传算法

针对目标函数由公式（47）给出的非线性整数规划，提出了遗传算法，并将该目标函数作为适应度函数。所提出的遗传算法采用基于随机性的选择过程、交叉、变异和精英保留等计算技术。所提出的遗传算法分为两个阶段：初始阶段和次级阶段。在初始阶段，执行 d 次独立运行，以初始化初始种群中的解。将各次运行中后续连续的解能够有效提高系统可靠性的解纳入初始种群。这种选择初始

种群将使解的演化朝正确方向发展，并提高遗传算法的效率。值得注意的是，不具备精英保留的常规遗传算法使用随机初始化的种群来演化解。

使用遗传算法对两个问题进行评估：问题空间较小的评估示例一和问题空间较大的评估示例二。对于评估示例二或大型问题，对所提出的遗传算法的初始阶段进行了额外修改，以突出所提出的遗传算法相较于常规遗传算法的有效性。具体讨论如下。

首先，为了最大化系统可靠性，在不考虑 x_{ij} 的情况下估算 A (a_{ij})。然后，根据方程 (53) 设置启动策略 x_{ij} 。通过在方程 (52) 中用 p_{ij} 替换 E_{ij}，可以确定不考虑启动策略时函数 j 的可靠性，结果如方程 (54) 所示。

$$r_{fj} = 1 - \prod_{i \in I_j} \left[1 - rd_i \left(1 - (1 - r_{ij}.p_{ij})^{a_{ij}}\right)\right] \quad (54)$$

其次，估计 A_{ij} 涉及两个步骤，具体如下。

1. 通过启用功能来选择可提高功能可靠性 (r_{fj}) 的组件，使最大时延处于C以内。2. 在两个OIC（例如a、b）之间进行选择：启用‘a’中的一个功能，并禁用‘b’中的相同功能，以使r_{fj}改进达到最大值。根据i_j，确定 A (a_{ij}) 并将其设置为 X，如方程 (53) 所示。

最后，将初始阶段获得的解纳入所提出的遗传算法次级阶段的初始种群中。

下面介绍所提出的遗传算法的次级阶段。

1. 初始化遗传算法参数：P_SIZE（种群规模）、P_MUTAT（变异概率）、P_CROSS（交叉概率）和 M_GEN（代数）。该问题的染色体编码如图 1.8 所示。染色体由决策变量 X（表示启动策略）和 U（表示所使用的各类型功能的数量）组成。二进制编码字符串中 1 的数量表示采用启动策略的功能（X），如图 1.8 所示。

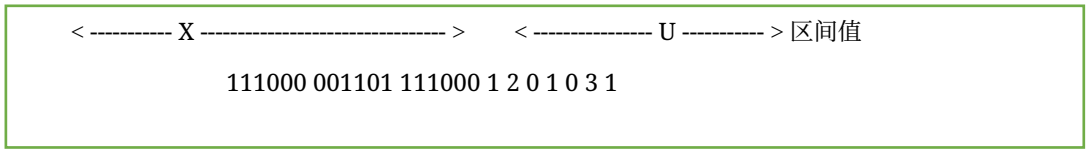


图 1.8 染色体编码示例

2. 对每条染色体进行适应度评估。适应度函数用于从种群中选择最佳染色体。对于 RAP 系统，将染色体的区间值目标函数值作为适应度函数值。只有适应度值大于阈值的染色体才会被纳入选择。

3.选择：主要目标是进化出更好的解，并从当前种群中淘汰低于平均水平的解，以进入下一代。在所提出的遗传算法中，根据个体的适应度值，从当前种群中随机选择个体。定义1.3 和 1.4 被适当地用于确定选择高质量染色体的可靠性区间。

当前版本为版本 1.0。 稿件正在审稿中。修改版或扩展版将刊登于期刊。

4. 以概率（P_CROSS）参与交叉的染色体/个体将用于产生后代。采用两点交叉算子。

5. 新物种不会突然产生。因此，选择变异概率 P_MUTAT 来创建新的种群。变异会增加种群的多样性。

6. 精英保留：在所有迭代次数中保留最优解，以提高遗传算法的性能。 .

7. 重复迭代，直到达到关于可靠性和成本的最大最优解。通过评估目标函数来确定系统可靠性。比较 j^{th} 和 $(j+1)^{th}$ 代时系统可靠性的最佳结果，并接受其中较优者。重复步骤 1 至 7，直到系统可靠性的变化变得不显著。

8.1 遗传算法评估

本文考察用于估计系统可靠性的 OSS 模型参数的初始化与评估。以下内容也适用于粒子群优化（PSO）的评估。(a) 准备度参数 rd_i：在包含 OIC 的多核系统中，运行初期组件级就绪度较高。因此，rd_i 设为 0.99。在后续请求中，评估系统可靠性时会考虑先前的准备度。(b) 唤醒概率 p_{ij}：在 OIC 中，已采用启动策略的功能，其唤醒概率设为 1。尚未采用启动策略的功能，其唤醒概率介于 0.1 和 0.9 之间。在某一时间段内于 OIC 上开启的功能，其唤醒概率将高于未能满足截止时间的其他功能。因此，同一功能在不同 OIC 上可能具有不同的唤醒概率。(c) 参数 r(t)（包含 r_{ij}）：由于 OIC 上的迁移和指令执行等操作存在不精确性，因此赋予区间值。表5中给出的区间值样本集取自 Roy 等人 [44] 的研究。该区间值集合已用于：(a) 研究系统熵 [44], ；以及 (b) 研究电力系统中的最大负荷点变化和电压稳定性 [31]。这组区间数的意义在于，它涵盖了所有类型的区间：互不相交、部分重叠，以及一个区间包含另一个区间。表5所列集合1中包含互不相交的区间数。表5所列集合2–4中包含互不相交和部分重叠区间数的混合。表5中的区间数集合 5 则包含一个区间数被另一个区间数包含的情况。这些区间数集合会使系统可靠性最大化的决策在乐观与悲观之间摆动。系统稳定性将接受现实条件下的检验，并在灵敏度分析部分进行考察。

考虑了两个评估问题——前文提到的“评估示例一”和“评估示例-二大型”。在评估示例一中，考虑了六项功能和三个 OIC。在评估示例二中，考虑了十项功能和六个 OIC。

8.2 评估示例一

在初始阶段，使用十次独立运行来初始化初始种群中的解。对于表5中所述的每个间隔数集合，在所提出的遗传算法的次级阶段执行20次独立运行，以评估包含六项功能和三个OIC的评估示例一。OIC 支持的六项功能为加法、移动、递增、递减、减法和除法。种群规模初始化为100，代数设置为200。使用两点交叉算子生成新后代。超过80%的种群执行两点交叉操作。6%的个体发生变异。唤醒概率和就绪参数列于表6中。组件i上功能j的成本 C_{ij}（以时钟周期计的执行时间）如表6所示。在公式（45）中，对于评估示例一，C设置为50个时钟周期。

选择函数、交叉和变异等遗传算法参数对于所提出的遗传算法和常规遗传算法均采用相似的设置。系统可靠性以系统可靠性（下界）和系统可靠性（上界）的形式给出的最佳结果列于表7中。所提出的遗传算法和常规遗传算法给出的最优解如表8所示。所提出的遗传算法提供了更优的系统可靠性（下限/上限）解

与常规遗传算法相比，适用于表8中的所有数据样本。由表8可知，在第一列中，X：101000 000010 101101，前六位表示第一个OIC中的ADD和INC功能是启动策略功能；接下来的六位表示第二个OIC中的SUB功能是启动策略功能；第三组六位表示第三个OIC中的ADD、INC、DEC和DIV功能是启动策略功能。由表8可知，在第一列中，U：2 2 2 3 3 3表示系统中可用的ADD、MOV、INC、DEC、SUB和DIV功能副本数量。对于表8中的给定X和U，所提出的遗传算法在C=44 时钟周期内达到的最优系统可靠性（下限/上界）为0.969002/0.970178。类似地，可以对表8中的其他条目进行解释。

从表7可以看出，对于表5中的所有样本，常规遗传算法和所提出的遗传算法给出的系统可靠性（上界/下界）值之间的差距更加明显。从表8中两种遗传算法给出的近似最优解之间的差异可以得出，初始阶段提高了所提出的遗传算法的效率。

表5 区间数（r_R, r_L）

区间 数字集合 **No.**	区间数类型	区间数格式为 **[**下界, 上界**]**
第 1 组	不相交区间	[0.68,0.72], [0.73,0.75], [0.78,0.81], [0.80,0.88], [0.89,0.95]
集合 2	不相交与部分重叠 重叠区间	[0.65,0.70], [0.71,0.73], [0.80,0.88], [0.82,0.87], [0.90,0.92]
集合 3	不相交且部分 重叠的区间	[0.60,0.67], [0.72,0.78], [0.78,0.83], [0.80,0.90], [0.87,0.956]
集合 4	不相交且部分 重叠区间	[0.64,0.65], [0.72,0.74], [0.80,0.88], [0.83,0.85], [0.90,0.95]
设置 5	其中包含的一个区间 其他、互不相交且部分 重叠区间	[0.63, 0.66], [0.64, 0.68], [0.65, 0.70], [0.65, 0.70], [0.73, 0.74], [0.75, 0.79], [0.76, 0.86],[0.75, 0.80], [0.77, 0.80], [0.75, 0.81], [0.78, 0.84], [0.80, 0.87], [0.88, 0.92], [0.89, 0.90],[0.91, 0.96]

表6 评估示例一的成本、唤醒概率和准备度参数

成本(C_{ij})	唤醒概率(p_{ij})	准备度 rd_i
4 5 4 1 1 35 4 5 4 1 1 35 4 5 4 1 1 35	0.98 0.9 0.9 0.96 0.87 0.87 0.82 0.82 0.82 0.9 0.9 0.9 0 0.9 0.9 0.9 0.9 0.9	rd =rd_1 = rd_2 = rd_3 = 0.99

当前版本是版本 1.0。 稿件正在审稿中。修改或扩展后的版本将发表在期刊上。

表7 使用遗传算法时，包含3个OIC和6项功能的评估示例一的最佳系统可靠性（下限/上限）

所提出的遗传算法	表6.1 (集合1)	表6.1（集合2)	表6.1（集合3）	表6.1（集合4）	表6.1（第 5 组）
系统可靠性（较低）（最佳）	0.969006	0.969328	0.968163	0.969328	0.969591
系统可靠性(上方)(最佳)	0.970178	0.969802	0.970178	0.970178	0.970237
常规遗传算法	表6.1 (第 1 组)	表6.1 (集合2)	表6.1 (集合3)	表6.1 (集合4)	表6.1（第 5 组）
系统可靠性(LOWER)(最佳)	0.968975	0.968452	0.967877	0.968454	0.962434
系统可靠性（上界）（最佳）	0.970132	0.969230	0.970132	0.969947	0.968746

表8 使用遗传算法求解含3个OIC和6项功能的评估示例一的最优解

所提出的遗传算法	表6.1 （集合1）	表6.1 (集合 2)	表6.1 （集合3）	表6.1 (集合 4)	表6.1 （第 5 组）
X	101000 000010 101101	110100 000111 011000	010000 000011 110111	111011 000110 001000	111110 001111 011110
U: **A** OICs	2 2 2 3 3 3 3	2 2 2 3 3 3 3	2 2 2 3 3 3 3	2 2 2 3 3 3 3	1 2 3 3 3 1 3
系统可靠性下限/上限	0.969002/ 0.970178	0.968437/ 0.969224	0.958130/ 0.966943	0.951516/ 0.963214	0.962376/ 0.968744
成本（周期）	44	40	46	49	41
常规遗传算法	表6.1 （集合1）	表6.1 （集合2）	表6.1 （集合3）	表6.1 （集合4）	表6.1 （第5组）
X	010111 000001 111111	111101 000111 101111	111101 000110 011111	011011 000111 101001	110110 000010 111101
U: A OICs	2 2 2 3 3 3 3	2 2 2 3 3 3 3	2 2 2 3 3 3 3	2 2 2 3 3 3 3	2 2 3 3 3 1 3
系统可靠性下限/上限	0.968780 0.970131	0.960412 0.964004	0.953624 0.967867	0.968101 0.968977	0.958415 0.967867
成本（以周期计）	41	49	49	45	49

当前版本是版本 1.0。 稿件正在审稿中。修改或扩展后的版本将发表在期刊上。

8.3 评估示例二

评估示例二包含 10 个功能和 6 个 OIC，通过 10 次独立试验进行评估，每次试验运行 60 次，并针对表5中的数据样本进行 20000 代。10 个功能分别记为 F1、F2、F3.....F10。唤醒概率、成本 C_{ij}（以及 C = 3000 周期）和就绪参数列于表 9 中。

评估示例二问题的最佳系统可靠性（下限/上限）结果列于表 10 中。所提出的遗传算法针对评估示例二得到的最优解见表 11。由表 11 可知，X：1 0 0 0 0 0 1 1 1 0，0 0 0 0 0 0 1 1 0 0，0 0 0 0 0 0 0 0 0 1，0 1 0 0 0 0 0 0 0 0，0 1 0 0 1 1 0 1 1 0，0 0 0 0 1 0 0 0 0 1，前十位表示在第一个 OIC 中对 F1、F7、F8 和 F9 采用了启动策略。其余位串可依此类推。由表 11 可知，U：3 3 3 1 3 3 3 3 4 4 表示系统中可用的功能 F1、F2、F3.....F10 的副本数量。对于表 11 中给定的 X 和 U，所提出的遗传算法在 C=2500 个时钟周期内达到的最优系统可靠性（下限/上限）为 0.989966 / 0.989997。表 11 中的其他条目也可依此解读。将所提出的遗传算法得到的系统可靠性解与常规遗传算法进行比较。根据表 10 和图 1.9，与常规遗传算法相比，所提出的遗传算法收敛更快。所提出的遗传算法稳定在 0.989998/0.990000，而常规遗传算法给出的结果为 0.989966/0.989997。所提出的遗传算法在大问题空间中收敛更快，是由于其初始阶段所做的改进，这一点在图 1.9 中显而易见。由表 11 可知，所提出的遗传算法给出的系统可靠性（下限/上限）为 0.989966 / 0.989997，优于常规遗传算法给出的 0.989876 / 0.989991。

表 9 评估示例二的成本、唤醒概率和就绪参数

成本 C_{ij}	100 100 250 250 500 500 600 600 800 800 100 100 0 250 500 500 600 600 800 800 100 100 250 0 500 500 600 600 800 800 100 100 250 250 500 500 600 600 800 800 100 100 250 250 500 500 0 600 800 800 100 100 250 250 500 500 0 600 800 800
唤醒 概率(p_{ij})	0.90.7 0.7 0.9 0.9 0.7 0.8 0.8 0.8 0.7 0.9 0.7 0 0.7 0.9 0.9 0.8 0.8 0.8 0.7 0.90.7 0.9 0 0.7 07 0.8 0.8 0.8 0.7 0.90.7 0.9 0.7 0.9 0.7 0.8 0.8 0.8 0.7 0.90.8 0.8 0.9 0.9 0.9 0 0.8 0.8 0.8 0.9 0.8 0.8 0.9 0.9 0.8 0 0.8 0.8 0.8
准备度 rd_i	$rd_1=rd_2 = rd_3=rd_4=rd_5=rd_6=0.99.$

采用 GA 的 6 个 OIC 和 10 项功能的评估示例二中，最佳系统可靠性（LOWER/上限）

所提出的 GA	系统可靠性（最佳）（LOWER）	系统可靠性（最佳）（上限）	常规遗传算法	系统可靠性（最佳）（LOWER）	系统可靠性(最佳) (上限)
表6.1 (第 1 组)	0.989998	0.990000	表6.1 (集合 1)	0.989966	0.989997
表6.1 (集合 2)	0.999983	0.999993	表6.1 (集合 2)	0.999833	0.999906
表6.1 (集合 3)	0.999923	0.999996	表6.1 (集合 3)	0.999887	0.999976
表6.1 (集合 4)	0.999968	0.999995	表6.1 (集合 4)	0.999857	0.999989
表6.1 (集合 5)	0.999989	0.999999	(集合 5)	0.999936	0.999953

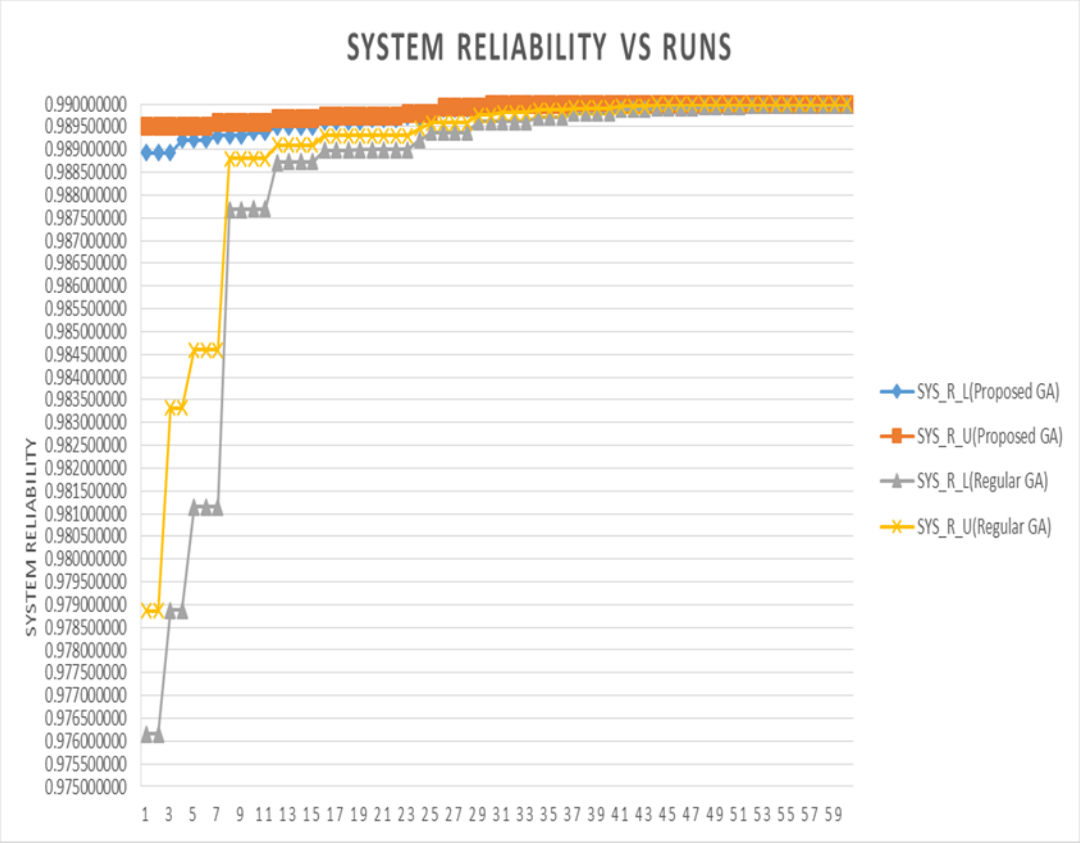


图 1.9 评估示例二的解演化

Table 11含6 个 OIC 和 10 项功能的评估示例二使用遗传算法的最优解

所提出的遗传算法	表6.1（第1组）	表6.1（集合 2）
X	1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 1 0 0 0 0 0 1 0 0 0 0 1	0 1 0 0 0 1 0 0 1 0 0 1 0 0 0 0 1 1 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0
U: **A** OICs	3 3 3 1 3 3 3 3 4 4 6	5 2 4 2 3 2 4 3 3 2 6
系统可靠性 下限/上限	0.989966 / 0.989997	0.999968/0.999984
成本（周期）	2500	1900
常规遗传算法		
X	1 0 0 0 0 1 1 1 0 0 0 1 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 1 1 0	0 0 0 1 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 0 0 0 0 1 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
U: A OICS	4 3 3 3 3 3 3 3 4 4 6	4 4 5 3 6 5 3 5 3 3 6
系统可靠性 下限/上限	0.989876 / 0.989991	0.999006 / 0.999646
成本（周期），	2500	1900

表 11（续）

所提出的遗传算法	表6.1(集合 3)	表6.1(集合 4)	表6.1（第5组）
X	0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0	0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 1 0 1	0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0
U: **A** OICs	3 3 2 2 3 3 3 2 3 3 6	4 3 3 3 3 2 3 3 3 4 6	2 3 3 3 3 3 4 3 3 3 6
系统可靠性 下限/上限	0.999602/0.999968	0.999903 / 0.999945.	0.999977 / 0.999997
成本（以周期计）	1900	2150	2000
常规遗传算法			
X	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 1 0 1	0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1	1 1 1 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 1 0 1 0 0 1 0 0 0 0 1 1 1 1 1 0 0
U: A OICS	3 3 2 2 4 3 3 4 3 3 6	4 3 3 3 3 2 3 3 3 3 6	3 3 3 4 3 3 4 3 3 3 6
系统可靠性 下限/上限	0.999791 / 0.999991	0.999860 / 0.999917	0.999851/0.999952
成本（周期），	2350	2200	2350

8.4 敏感性分析

进行敏感性分析以研究遗传算法参数对目标函数的影响。该分析还有助于研究在目标函数中使用区间值数表示不精确性时系统的稳定性。遗传算法参数值也通过此分析选取。这里通过图1.10–1.29中的图表，展示了所提出的遗传算法在所有区间数（表5）下相对于遗传算法参数（P_SIZE、P_MUTAT、P_CROSS 和 M_GENER）的稳定性。从图1.10–1.29可以看出，所提出的遗传算法稳定且不变，从而为所提出的 RAP公式化提供高质量的解。下面列举了与遗传算法参数选择及其一致性相关的几项重要观察。

(a) 图1.10、1.14、1.18、1.22和1.26中的系统可靠性（下限/上限）随代数单调递增。参数 M_GENER = 200 是一个合适的值，能够反映表5所列所有区间数的系统可靠性（下限/上限）的稳定性。

(b)P_SIZE（种群规模）相对于图 1.11、1.15、1.19、1.23 和1.27中的系统可靠性（下限/上限）进行变化。值得注意的是，所提出的遗传算法采用了随机选择。即便如此，系统可靠性（下限/上限）仍随着种群规模的增加而成比例提高，并在P_SIZE =100 时趋于稳定。当种群规模为100 或更大时，P_SIZE 不会影响我们所提出的遗传算法的最优解。

(c) P_CROSS：图 1.12、1.16、1.20、1.24 和 1.28 展示了交叉对系统可靠性的影响。从这些图中可以看出，交叉率的增加不会对系统可靠性（下限/上限）产生任何影响。交叉率的增加不会导致系统可靠性值发生异常变化。

(d) P_MUTAT：较高的变异水平会产生更多的遗传组合，有时确实会对系统产生影响。图 1.13、图1.17、图1.21和图1.29反映了系统可靠性的稳定性。在所提出的遗传算法中，较高的变异率与精英保留相结合，可使种群在后续过程中更好地演化出解，并恢复前几代中丢失的解。为 RAP 选择的 P_CROSS 和 P_MUTAT 比率适合在较大范围内维持稳定性。

(e) 组件可靠性所用区间数的类别选择并不重要，因为对于 M 和 N，只能存在一个最优系统。敏感性分析表明，区间值数的选择不会影响系统的最优解。系统可靠性随部署的组件和功能数量增加而单调递增。因此，所提出的遗传算法得到的最优解不会受到影响。由此确认了系统的稳定性。

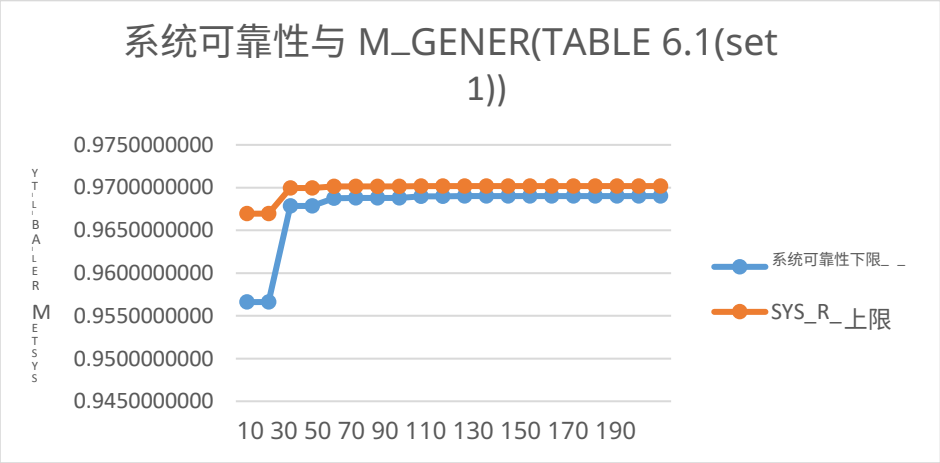


图1.10 表5中区间数集合1的系统可靠性与代数的关系。

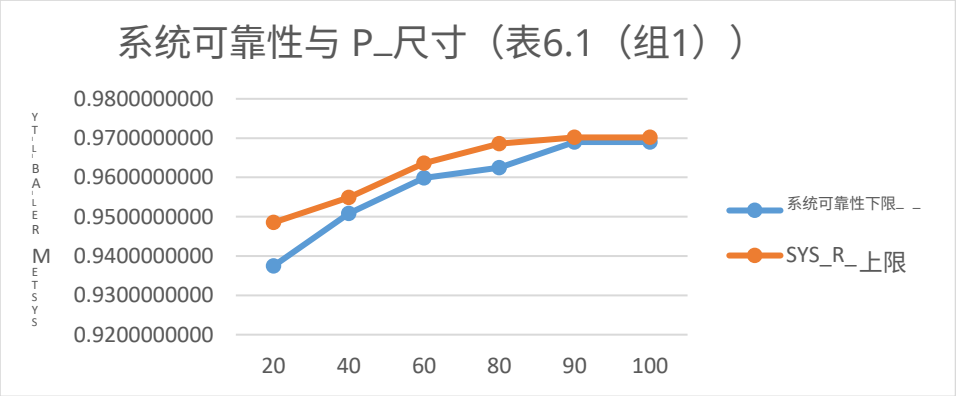
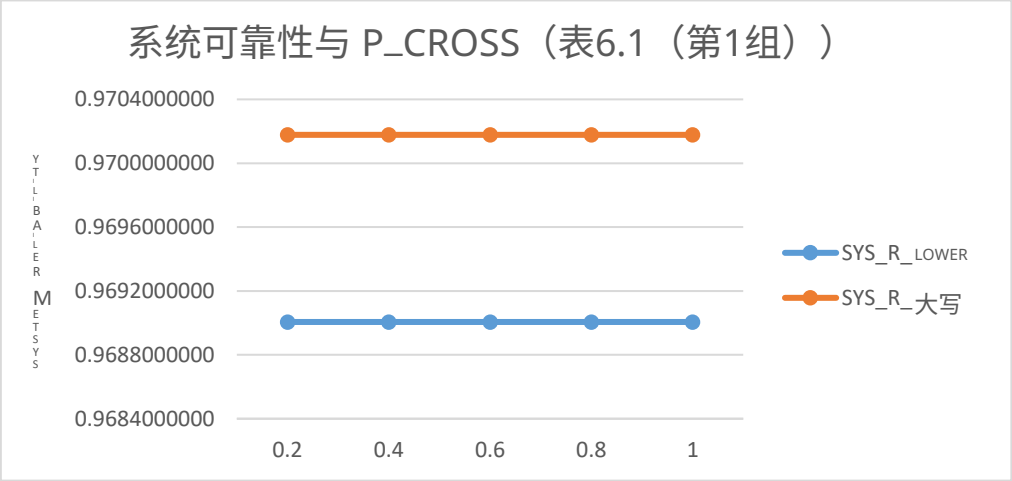


图1.11 表5中区间数集合1的系统可靠性与种群规模的关系



Fig图1.12 表5中区间数集1的系统可靠性与交叉概率关系

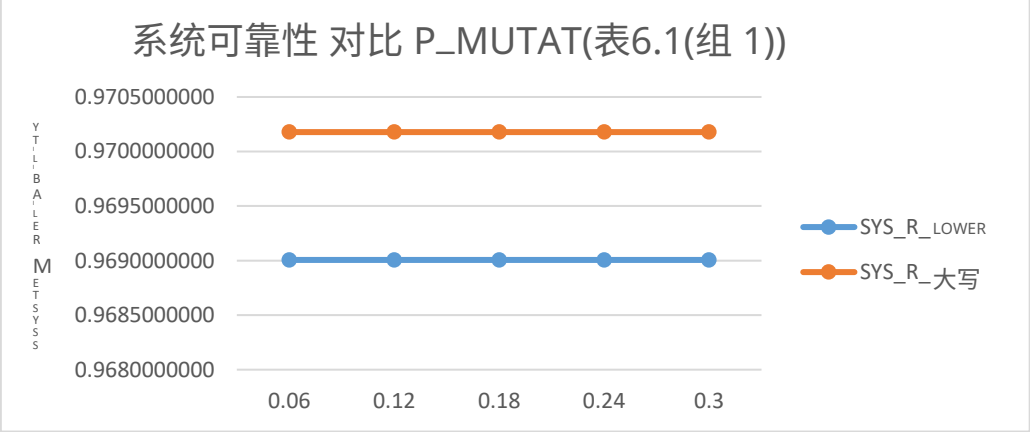


图1.13 表5中区间数集1的系统可靠性与变异概率的关系

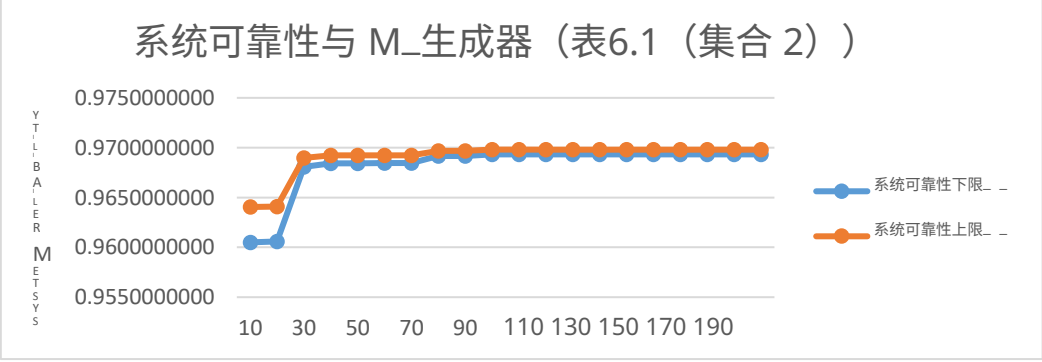


图1.14 表5中区间数集合 2 的系统可靠性与代数的关系

当前版本是版本 1.0。 稿件正在审稿。修改或扩展后的版本将在期刊中发表。

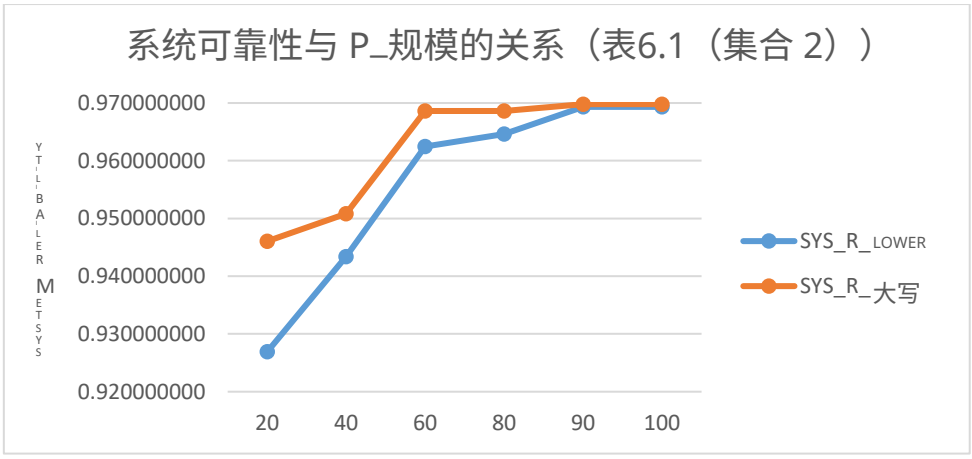


图1.15 表5中区间数集合 2 的系统可靠性与种群规模的关系

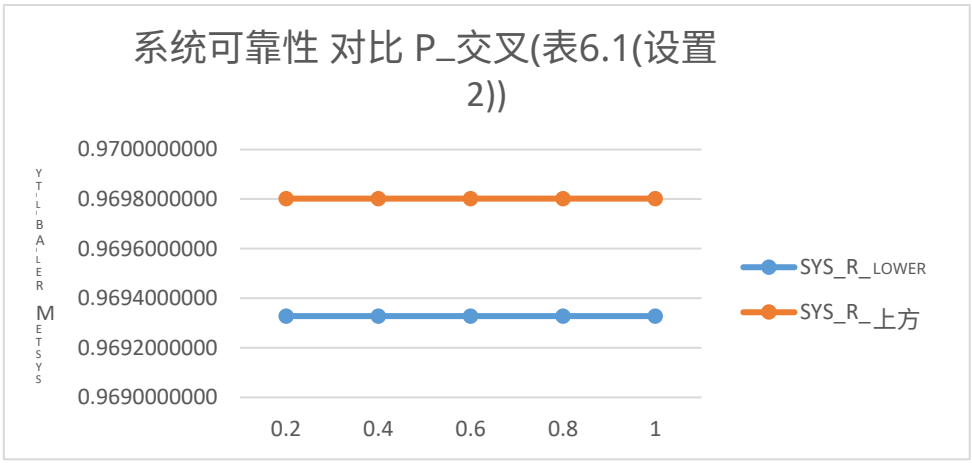
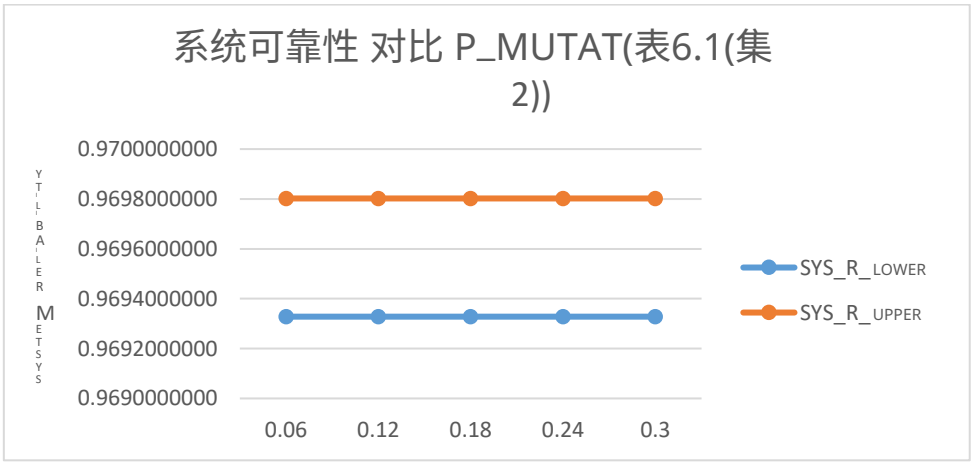


图1.16 表5中区间数集合 2 的系统可靠性与交叉概率的关系



F图1.17 表5中区间数集合 2 的系统可靠性与变异概率的关系

当前版本是版本 1.0。 稿件正在审稿。修改或扩展后的版本将在期刊中发表。

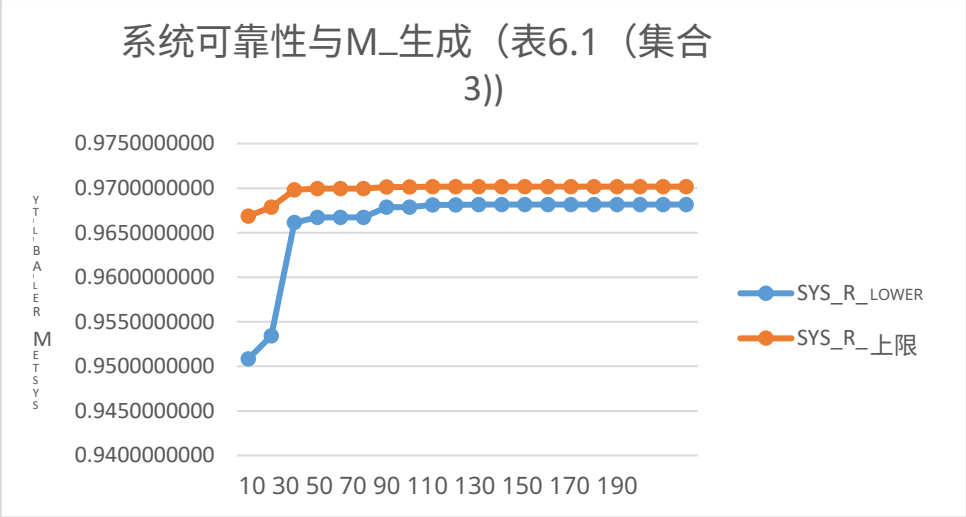


图1.18 表5中区间数集合3的系统可靠性随代数变化

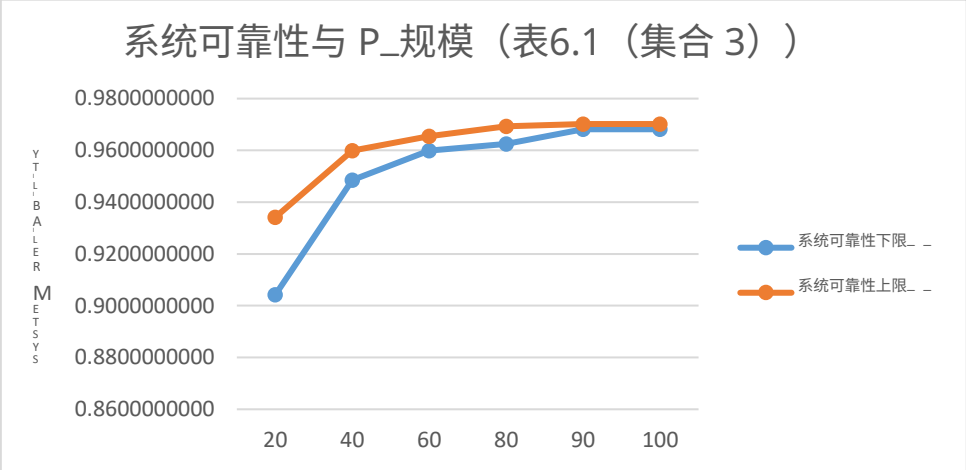


图1.19 表5中区间数集合3的系统可靠性随种群规模变化

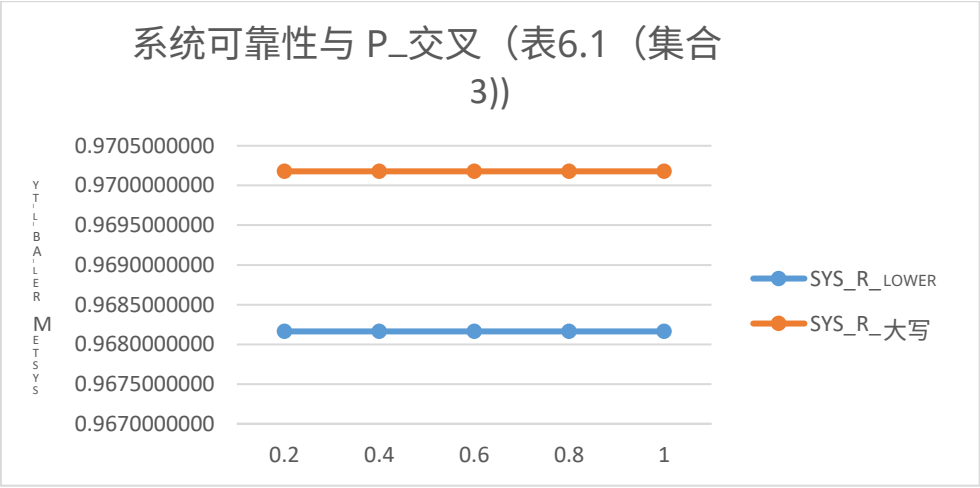


图1.20 表5中区间数集合3的系统可靠性随交叉概率变化

当前版本是版本 1.0。 稿件正在审稿。修改或扩展后的版本将在期刊中发表。

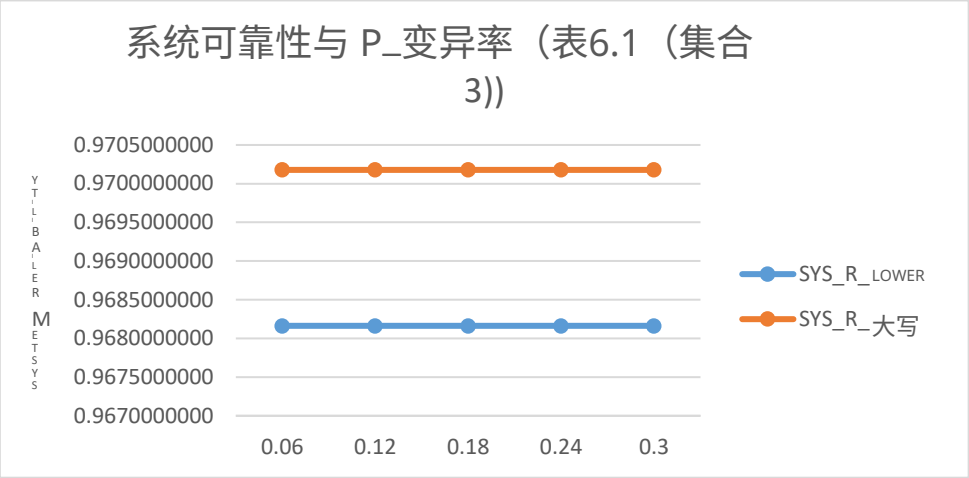


图1.21 表5中集合3的区间数的系统可靠性与变异概率的关系

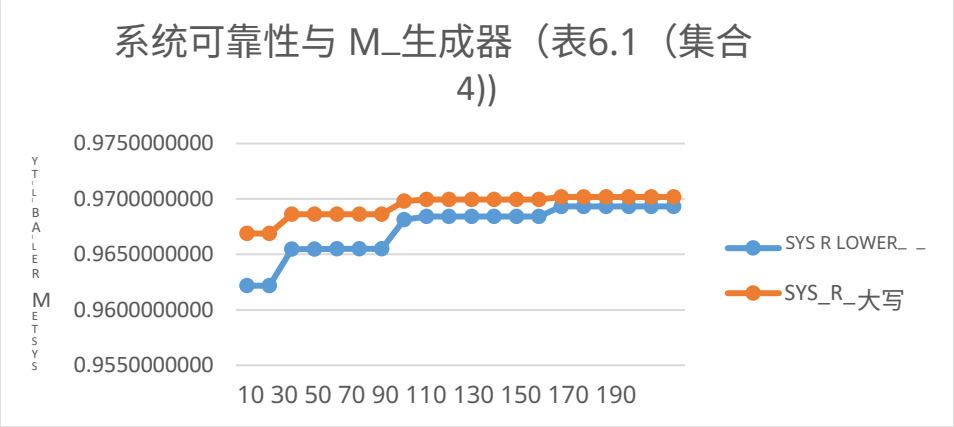


图1.22 表5中集合4的区间数的系统可靠性与代数的关系

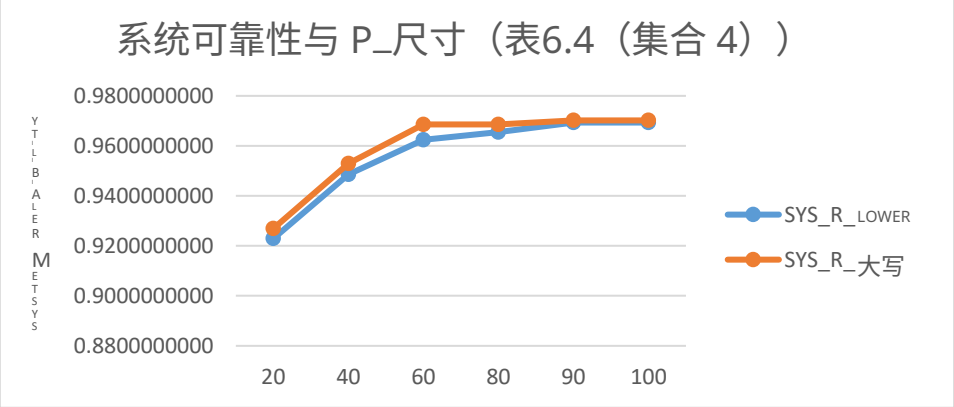
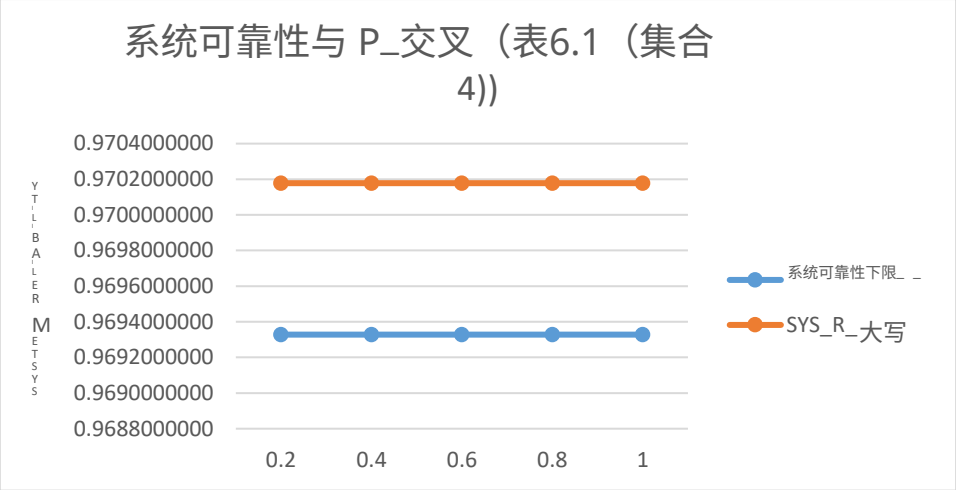


图1.23 表5中集合4的区间数的系统可靠性与种群规模的关系

当前版本为版本 1.0。 稿件正在审稿。修改或扩展后的版本将发表于该期刊。



F图1.24 表中区间数集合 4 的系统可靠性与交叉概率的关系 **5**

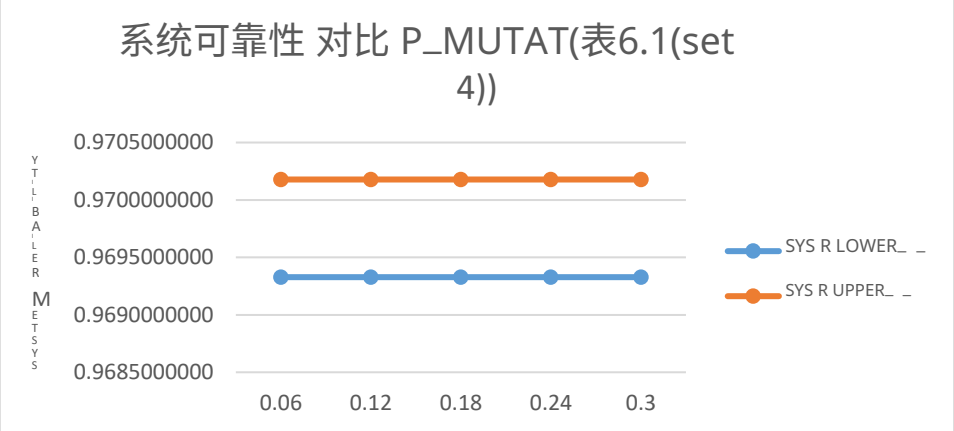


图1.25 表5中区间数集合 4 的系统可靠性与变异概率的关系

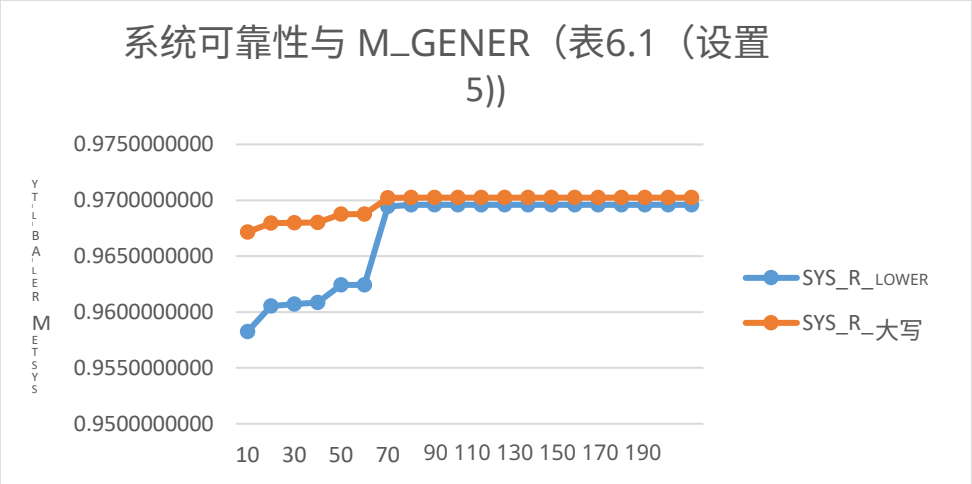


图1.26 表5中区间数集合 5 的系统可靠性与代数的关系

当前版本为版本 1.0。 稿件正在审稿中。修改或扩展版本将刊登于期刊。

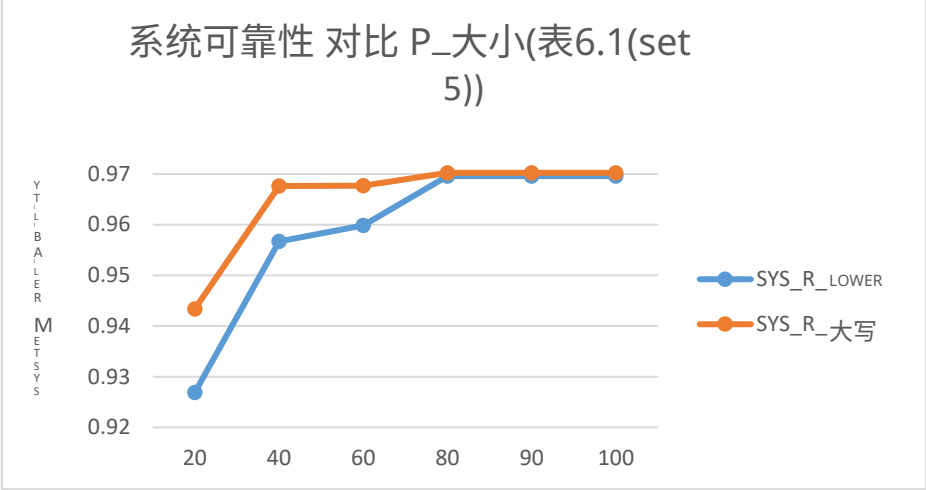
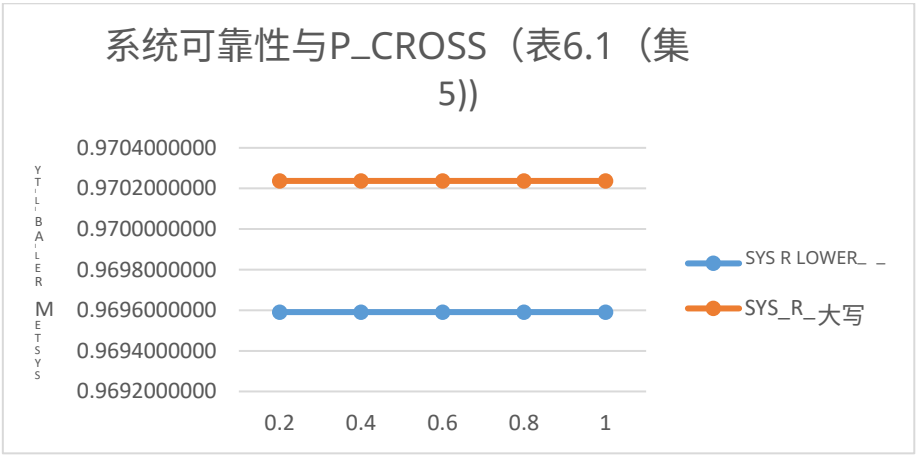


图 1.27 表5中区间数集合 5 的系统可靠性与种群规模的关系



F图 1.28 表中区间数集合 5 的系统可靠性与交叉概率 5

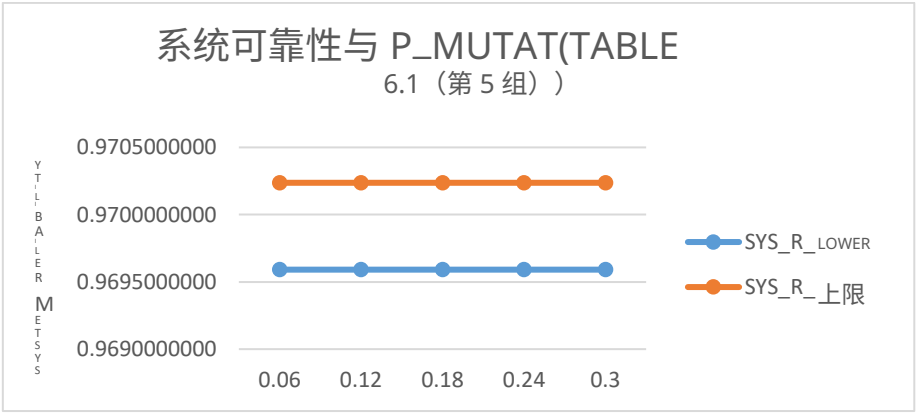


图 1.29 表5中区间数集合 5 的系统可靠性与变异概率关系

当前版本是版本 1.0。 稿件正在审稿中。修改或扩展后的版本将刊载于期刊。

9. 区间 RAP 的粒子群优化

粒子群优化（PSO）是解决大型非线性优化问题的一种有效替代方法。针对区间 RAP，提出了粒子群优化方法，并将公式（47）中的目标函数作为适应度函数。粒子群优化以鸟群的集群行为为概念基础，构建搜索框架。与遗传算法仅执行全局搜索不同，这一概念框架有助于为优化问题寻找局部和全局解。在粒子群优化中，会维护一个由多个粒子组成的群体，每个粒子都代表给定优化问题的一个候选解。所有粒子都会遍历搜索空间，并根据邻近粒子的信息及自身过去的位置调整其位置。设 x_i^t 为搜索空间中粒子 i 在时间步 t 的位置，则粒子的移动由以下方程（55）决定。

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (55)$$

在我们所提出的粒子群优化中，粒子位置映射为 r (t)，它是一个区间数，方程 (55) 可重述为方程 (56) 和 (57) 所示的形式。

$$x_{i(R)}^{t+1} = x_{i(R)}^t + v_{i(R)}^{t+1} \quad (56)$$

$$x_{i(L)}^{t+1} = x_{i(L)}^t + v_{i(L)}^{t+1} \quad (57)$$

$x_{i(R),}^{\ t}$ $x_{i(L),}^{\ t}$ $[x_{min}, x_{max}]$ 其中 x_{min}, x_{max} 是 $[r_R, r_L]$ 区间值，列于表5中。$v_{i(R)}^t$, $v_{i(L)}^t$ 是群体中第 i 个粒子对在时间 t 的速度。所有粒子朝已知较优解移动所产生的认知影响和社会影响由速度参数控制。$x_{i(R),}^{\ t}$ $x_{i(L),}^{\ t}$ 粒子对的位置通过迭代累加速度进行更新，并用于计算目标函数。

下面介绍用于更新速度的两种粒子群优化变体：全局最优粒子群优化（Gbest）和局部最优粒子群优化（Lbest）。在全局最优粒子群优化方法中，群体中所有粒子之间的最优位置会影响每个粒子的位置。采用星型拓扑从群体中的所有粒子收集已知的最优解。局部最优粒子群优化方法允许每个粒子的位置由邻域中粒子的已知最优位置控制，这类似于环形拓扑。在这两种方法中，每个粒子对都有一个当前位置 $x_{i(R,L)}^t$、当前速度 $v_{i(R,L)}^t$ 以及个体最优位置 $P_{best,\,i(R,\,L)}$。个体最优位置表示从初始化到时间 t 期间，粒子对 i 根据最大化问题的目标函数所确定的最大值。所有 $P_{best,\,i(R,\,L)}$ 中的最大值称为全局最优位置，记为$G_{best(R,\,L)}$。在局部最优粒子群优化中，$L_{best,\,i(R,\,L)}$是粒子对 i 从初始化到时间 t 期间获得的最优解或最优位置。$P_{best,\,i(R,\,L)}$、$G_{best(R,\,L)}$、$L_{best,\,i(R,\,L)}$根据定义1.3（第 III 节中提到）进行更新，用于最大化问题的乐观型决策。

用于控制的方程中，Gbest PSO 中粒子对 i 的速度控制方程见方程 (58) 和 (59)).

$$v_{i(R)}^{t+1} = \omega v_{i(R)}^t + \varphi_1 r_1^t \left[P_{best,i(R)}^t - x_{i(R)}^t\right] + \varphi_2 r_2^t [G_{best(R)} - x_{i(R)}^t] \quad (58)$$

$$v_{i(L)}^{t+1} = \omega v_{i(L)}^t + \varphi_1 r_1^t \left[P_{best,i(L)}^t - x_{i(L)}^t\right] + \varphi_2 r_2^t [G_{best(L)} - x_{i(L)}^t] \quad (59)$$

Lbest PSO 方法中粒子对 i 的速度控制方程见方程 (60) 和 (61)。

$$v_{i(R)}^{t+1} = \omega v_{i(R)}^t + \varphi_1 r_1^t \left[P_{best,i(R)}^t - x_{i(R)}^t\right] + \varphi_2 r_2^t [L_{best,i(R)} - x_{i(R)}^t] \quad (60)$$

$$v_{i(L)}^{t+1} = \omega v_{i(L)}^t + \varphi_1 r_1^t \left[P_{best,i(L)}^t - x_{i(L)}^t\right] + \varphi_2 r_2^t [L_{best,i(L)} - x_{i(L)}^t] \quad (61)$$

定义 1.1 和 1.2 中提到的区间算术用于求解方程 (58)、(59)、(60) 和 (61)。在方程 (58)、(59)、(60) 和 (61) 中，此处采用了 $\boldsymbol{\omega}$ 随时间变化的惯性权重，发现它能线性地提升性能，并控制先前速度对更新后速度的影响。

下面详细说明方程 (58)、(59)、(60) 和 (61) 中使用的所有参数。

$G_{best(R,\,L)}$：是粒子对 i 从初始化到时间 t 的全局最优位置；

$L_{best,\ i(R,\,L)}$: i 是粒子对 i 从初始化到时间在邻域中的最优位置 t;

$P_{best,\ i(R,L)}$：是粒子对 i 从初始化到时间 t 的个体最优位置；

$v_{i(R,L)}^{t}$：粒子对 i 在时间 t 的速度；

$x_{i(R,L)}^{t}$：是粒子 i 在时间 t 的位置；

ω：是惯性权重，等于 $(w_1 - w_2)\ ((T_{max} - t)/T_{max}) + w_2$，，其中 w1 和 w2 分别是初始权重和最终权重，t 是当前迭代次数，T_{max} 是迭代次数。

$\varphi_1\ and\ \varphi_2$：正加速度系数，也称为认知权重和社会权重；

$r_1^t\ and\ r_2^t$：时间 t 处均匀分布 U (0,1) 中的随机数。

最后，利用速度更新所有粒子对的位置。对每个粒子对计算目标函数，并据此检查解的收敛情况。重复这些步骤，以在下一次迭代中计算 $P_{best,\ i(R,\ L)}$、$G_{best(R,\ L)}$、$L_{best,\ i(R,\ L)}$。所提出的 Gbest/Lbest 粒子群优化算法的逐步说明如下。

9.1 所提出的 Gbest/Lbest 粒子群优化算法的逐步说明

针对区间 RAP 适当修改后的 Gbest/Lbest 粒子群优化算法实现如下。

1. 初始化粒子（对）种群。使用目标函数评估每个粒子对。初始时，粒子的 $P_{best,i(R,L)}$（个体最优）就是该粒子本身。从已知规模的粒子池中选取一组粒子对。将表5中提到的区间数分配给粒子对，使系统可靠性最大化。

2. a. $G_{best(R,L)}$ 根据所有粒子对进行评估。

2. b. $L_{best,\ i(R,L)}$ 根据邻域中的粒子进行评估。

3. a. 使用方程 (58)、(59) 计算 Gbest 粒子群优化中粒子对的速度。

3. b. 使用方程 (60)、(61) 计算 Lbest 粒子群优化中粒子对的速度。

4. 归档每个粒子更新后的速度。

5. $P_{最优,\ i(R,\ L)}$、$G_{best(R,\ L)}$、$L_{最优,\ i(R,\ L)}$根据定义 1.3，针对最大化问题的乐观型决策进行更新。

6. 迭代计数器递增。重复步骤 3 和 4，并在解收敛时终止算法。

9.2 所提出的全局最优粒子群优化／局部最优粒子群优化算法的通用代码片段。

1. 粒子位置 $x_{i(R,L)}^{0}$、速度 $v_{i(R,L)}^{0}$、加速度系数 φ_1, φ_2 均已初始化。使用方程 6.14 计算时间 t =0 处的 $x_{i(R,L)}^{0}$，并据此计算 $f_{i(R,L)}^{0}$。粒子数 (P) 和迭代次数 (N) 已适当设定。

2. $r_1^t\ and\ r_2^t$ 进行随机初始化。

重复

3.a. $v_{i(R)}^{t+1} = \omega v_{i(R)}^{t} + \varphi_1 r_1^t\left[P_{best,i(R)}^{t} - x_{i(R)}^{t}\right] + \varphi_2 r_2^t[G_{best(R)} - x_{i(R)}^{t}]$ 和 $v_{i(L)}^{t+1} = \omega v_{i(L)}^{t} + \varphi_1 r_1^t\left[P_{best,i(L)}^{t} - x_{i(L)}^{t}\right] + \varphi_2 r_2^t[G_{best(L)} - x_{i(L)}^{t}]$ 用于在全局最优算法中更新速度。

3. b. $v_{i(R)}^{t+1} = \omega v_{i(R)}^{t} + \varphi_1 r_1^t\left[P_{best,i(R)}^{t} - x_{i(R)}^{t}\right] + \varphi_2 r_2^t[L_{best,i(R)} - x_{i(R)}^{t}]$ 和$v_{i(L)}^{t+1} = \omega v_{i(L)}^{t} + \varphi_1 r_1^t\left[P_{best,i(L)}^{t} - x_{i(L)}^{t}\right] + \varphi_2 r_2^t[L_{best,i(L)} - x_{i(L)}^{t}]$ 用于更新局部最优算法中的速度。

3. c. $r_1^t\ and\ r_2^t$ 在每次迭代中针对每个粒子对随机生成。

4. $x_{i(R,L)}^{t+1} = x_{i(R,L)}^{t} + v_{i(R,L)}^{t+1}$

当 (i < P)

5. 计算每个粒子 $x_{i(R,L)}^{t}$的目标函数 $f_{i(R,L)}^{t}$。检查解对的收敛情况。重复这些步骤，直到解收敛。

6. 如果 t < N，则转到步骤 2。

9.3 粒子群优化的评估

如前所述，评估考虑了两个问题——评估示例一和评估示例二。在评估示例一中，考虑了六个功能和三个 OIC。在评估示例二中，考虑了 10 个功能和六个 OIC。用于估计系统可靠性的OSS 的初始化和模型参数评估，与遗传算法评估中的使用方式相同。

在评估示例一中，目标函数（方程47）考虑了由三个OIC 组成的多核系统的区间 RAP，其中每个 OIC 执行六项功能，并使用区间 –局部最优和区间 –全局最优粒子群优化算法进行评估。其种群规模为 30，存档规模为15，迭代次数为50。φ_1、 φ_2 、w_1 和 w_2 的值分别取为0.99876、0.99678、0.8999和 0.2466。唤醒概率和就绪参数列于表6中。成本 C_{ij}，即第i 个OIC上功能j的时钟周期执行时间，如表6所示。在方程45中，对于评估示例一，C 设置为50个时钟周期。区间–局部最优和区间 –全局最优粒子群优化算法给出的最优解列于表12中。显然，从表12可以看出，对于评估示例一问题，所提出的区间–局部最优和区间 –全局最优粒子群优化算法均给出了优于所提出的遗传算法和常规遗传算法的解。

表 12 使用粒子群优化时，包含 3 个 OIC 和 6 项功能的评估示例一的最优解

描述	表6.1 (第 1 组)	表6.1 (集合 2)	表6.1 （集合 3）	表6.1 （集合 4）	表6.1 （第 5 组）
X	101000 000010 101101	110100 000111 011000	010000 000011 110111	111011 000110 001000	111110 001111 011110
U: **A** OICs	2 2 2 3 3 3 3	2 2 2 3 3 3 3	2 2 2 3 3 3 3	2 2 2 3 3 3 3	2 2 3 3 3 2 3
区间全局最优 ***PSO*** 系统 可靠性 下限/上限	0.970276/ 0.970299	0.970281/ 0.970299	0.969060/ 0.970377	0.970274/ 0.970299	0.970276/ 0.970281
区间 局部最优 ***PSO*** 系统 可靠性 下限/上限	0.969423/ 0.970289	0.968900 0.970282	0.968960/ 0.970282	0.969440/ 0.970283	0.970189/ 0.970281
成本（周期数）	44	40	46	49	41

在评估示例二中，目标函数（公式（47））考虑了由六个OIC 组成的多核系统的区间 RAP，其中每个OIC 执行十项功能，并使用区间– Lbest 和区间–Gbest粒子群优化算法进行评估。其种群规模为 50，存档规模为15，迭代次数为100。φ_1、 φ_2 、w_1 和 w_2 的值分别取为 1.69876、0.19678、0.20和 0.10。这10项功能分别обознач为 F_1, F_2, F_3,...、F_{10}。唤醒概率、成本 C_{ij}（以及 C = 3000 周期）和就绪参数列于表 9 中。区间–Lbest 和区间 – Gbest粒子群优化算法给出的最优解列于表13 和表 14 中。在两个评估示例中，Gbest 粒子群优化均比 Lbest 粒子群优化收敛到更优的解。

最后，根据遗传算法和粒子群优化可知，随着 OIC 数量的增加，MCS - OIC 仍具有较强的可靠性。本文的结论将在下一节中给出。

使用粒子群算法评估含 6 个 OIC 和 10 项功能的评估示例二的表 13 最优解 **O**

描述	表6.1（集合 1）	表6.1（集合 2）
X	1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 1 0 0 0 0 0 1 0 0 0 0 1	0 1 0 0 0 1 0 0 1 0 0 1 0 0 0 0 1 1 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0
U: **A** OICs	3 3 3 1 3 3 3 3 4 4 6	5 2 4 2 3 2 4 3 3 2 6
区间全局最优粒子群优化 系统可靠性 下限/上限	0.999977/0.999997	0.999945/0.999984
区间局部最优粒子群优化 系统可靠性 下限/上限	0.999778/0.999913	0.999680/0.999883
成本（周期数）	2500	1900

当前版本为版本 1.0。 稿件正在审稿中。修改版或扩展版将刊登于期刊。

表14使用粒子群对含 6 个 OIC 和 10 项功能的评估示例二的最优解 **O**

描述	表6.1（集合 3）	表6.1(集合 4)	表6.1(组5)
X	0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0	0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 1 0 1	0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0
U: **A** OICs	3 3 2 2 3 3 3 2 3 3 6	4 3 3 3 3 2 3 3 3 4 6	2 3 3 3 3 3 4 3 3 3 6
区间全局最优粒子群优化 系统可靠性 下限/上限	0.999984/0.999998	0.999955/0.999980	0.999989/0.999997
区间局部最优粒子群优化 系统可靠性 下限/上限	0.999235/0.999766	0.999404/0.999592	0.999846/0.999947
成本（周期）	1900	2150	2000

10. 结论

(a) MCS – OIC 是一种新颖、经济且基于冗余的多核系统容错解决方案。本文使用 ASIC 仿真分析了不同配置的 MCS-OIC 的功耗和面积。由一个 MIPS 核心 + 和一个 OIC 组成的基本 MCS-OIC 模型，功耗为 1.1554W，面积为 306283μm^2。结果表明，其最大功耗开销和最大面积开销分别为 0.46% 和 11.4%，与其他方案相比相对较低。本文对由温备冗余 OIC 组成的多核系统的可靠性进行了建模和分析。使用单次系统对 OIC 的行为进行建模，并将 OSS 的模型参数映射到 MCS-OIC 的三种不同配置中。推导出了 MCS-OIC 的系统可靠性和 MTTF，并进一步推导了四种特殊情况下的系统可靠性。

(b) 使用爱尔朗分布的可靠性函数，研究了由两个传统核心和两个、三个、四个 OIC，以及由三个传统核心和三个、四个、五个 OIC 组成的系统的系统可靠性。结果观察到，随着 OIC 数量增加，系统可靠性也会提高。也就是说，对于具有 L 个传统核心和 M 个 OIC 的系统（M > L），可以实现更高的系统可靠性。因此，1:N 模型提供了一种可扩展的设计方案。

(c)使用两阶段遗传算法求解区间RAP。获得了评估示例一和评估示例二的最优解。结果表明，所提出的遗传算法比常规遗传算法演化出更优的解。已证明，系统可靠性随着功能级和组件级冗余的增加而单调提升。这是所提出的区间 RAP 的固有特性。据了解，该特性确保了具有 OIC 的多核系统的系统可靠性得到提升。

当前版本是版本 1.0。 稿件正在审稿中。修改或扩展版本将刊登于期刊。

(d)敏感性分析结果表明系统稳定性良好。遗传算法参数被固定，从而通过敏感性分析获得了最大化系统可靠性的更优解。据观察，遗传算法参数的变化不会影响系统稳定性和最优解。

(e)采用粒子群优化的两种变体——全局最优（Gbest）粒子群优化和局部最优（Lbest）——来求解区间 RAP。对于评估示例一，结果表明，所提出的 GBest和Lbest相较于所提出的遗传算法能够提供更好的结果。在评估示例二中，观察到粒子群优化变体与遗传算法所给出的解之间差异不显著。粒子群优化给出的结果还表明，系统可靠性随着功能级和组件级冗余的增加而提高。

(f) 通过对区间RAP进行遗传算法和粒子群优化评估可推断，系统是稳定的，且最优解不会因区间数而改变，区间 RAP 的单调性确保了系统可靠性的提升。

参考文献

研究论文的引文按照其在稿件中出现的顺序排列。

1. Moore, GE 2006a，《在集成电路上集成更多元件》，IEEE固态电路学会通讯，第20卷，第3期，第33–35页。

2. Moore, GE 2006b，《光刻与摩尔定律的未来》，IEEE固态电路学会通讯，第11卷，第3期，第37–42页。

3. Pollack, FJ 1999，《CMOS工艺技术未来几代中的新微体系结构挑战》，第三十二届IEEE/ACM微体系结构国际年会论文集，第2页。

4. Hennessy, JL & Patterson, DA 2007，《计算机体系结构：定量方法》，第四版。爱思唯尔。

5. Stefanos Kaxiras & Margaret Martonosi 2008，《计算机体系结构能效技术》。计算机体系结构综合讲座，Morgan Kaufmann。

6. Alghareb, FS, Ashraf, RA & DE Mara, RF 2017，《在能耗与稳健性的连续谱上设计和评估基于冗余的软错误屏蔽》，IEEE可持续计算汇刊，第3卷，第3期，第139–152页。

7. Guar, A & Mahmoodi, H 2012，《纳米级互补金属氧化物半导体中技术缩放对多米诺逻辑性能的影响》，第二十届 IEEE/IFIP 超大规模集成电路与片上系统国际会议论文集，第 295-298 页。

8. Mourad, S & Zorian, Y 2000，《电子系统测试原理》，约翰·威立父子出版公司。

9. Mukherjee, S 2011，《软错误架构设计》，Morgan Kaufmann。

10. Borkar, S 2005，《由不可靠组件构建可靠系统：晶体管变异性与退化带来的挑战》，IEEE Micro，第 25 卷，第 6 期，第 10-16 页。

11. Shivakumar, P、Kistler, M、Keckler, SW、Burger, D 和 Alvisi, L，2002，《技术趋势对组合逻辑软错误率的影响建模》，《IEEE 可信系统与网络国际会议论文集》，第 389-398 页。

12. Li, T、Ambrose, JA、Ragel, R 和 Parameswaran, S，2016，《软错误处理器设计：挑战与最新进展》，《ACM Computing Surveys》，第49卷，第3期，第57页（1-44）。

13. Gaisler, J2002，《一种基于 SPARC v8 架构的便携式容错微处理器》，IEEE 可信系统与网络国际会议论文集，第 409-415 页。

14. Lin、S、Kim、YB 和 Lombardi、F，2010 年，《纳米级 CMOS 抗辐射锁存器的设计与性能评估》，《IEEE 超大规模集成系统汇刊》，第 19 卷，第 7 期，第 1315–1319 页。

15. Slayman, CW2005，《面向陆地服务器和工作站的高速缓存与内存错误检测、纠正和减少技术》，《IEEE器件与材料可靠性学报》，第5卷，第3期，第397-404页。

16. Austin, TM 1999，'DIVA：一种适用于深亚微米微体系结构设计的可靠基底'，第三十二届 IEEE/ACM 微体系结构国际研讨会论文集，第 196–207 页。

17. Stackhouse, B、Bhimji, S、Bostak, C、Bradley, D、Cherkauer, B、Desai, J、Francom, E、Gowan, M、Gronowski, P、Krueger, D 和 Morganti, C 2008，'65 nm、20 亿晶体管四核 Itanium 处理器'，IEEE 固态电路期刊，第 44 卷，第 1 期，第 18–31 页。

18. Shashikiran Venkatesha 和 Ranjani Parthasarathi 2019，'32 位单指令核：一种面向多核系统的低成本、可靠且容错的核心'，ASTM 测试与评估期刊。https://doi.org/10.1520/ JTE20180492。ISSN 0090-3973。

19. Birnbaum, ZW、Esary, JD 和 Saunders, SC，1961，《多组件系统及其结构与可靠性》，《Technometrics》第3卷第1期，第55-77页。

20. Amari, SV、Zuo, MJ 和 Dill, G，2008，《用于分析可修复和不可修复 k-out-of-n: G 系统的 O (kn) 算法》，载于《Handbook of Performability engineering》（第309-320页），施普林格，伦敦。

21. Agarwal, M，1985，《应用于两单元备用系统随机分析的嵌入式半马尔可夫过程：两种故障类型》，《Microelectronics Reliability》第25卷第3期，第561-571页。

22. Ahmad, SH，1985，《具有 Weibull 故障的三单元冷备份系统的可靠性》，《Microelectronics Reliability》第25卷第2期，第325-330页。

23. Gnedenko, BV, Belayaev, YK & Solovyeb, AD 1969，《可靠性理论的数学方法》，Academic Press。

24. Li, X, Yan, R & Zuo, MJ 2009，'对具有比例风险率组件的温备用系统进行评估'，Operations Research Letters，第 37 卷，第 1 期，第 56-60 页。

25.Amari, V & Dill, G 2010，'采用温备用冗余的冗余优化问题'，Annual IEEE Reliability and Maintainability Symposium 论文集，第 1-6 页。

26. Huang, L & Xu, Q 2010，'表征具有核心级冗余的众核处理器的寿命可靠性'，IEEE/ACM International Conference on Computer-Aided Design 论文集，第 680-685 页。

27. Bierbaum RL，'核武器"固有可靠性"的概念模型'，2010，Annual Reliability and Maintainability Symposium 论文集，第 1-6 页，doi: 10.1109/RAMS.2010.5447992。

28. Xu, Y & Liao, H 2016，《包含多功能组件的一次性系统的可靠性分析与冗余分配》，IEEE《可靠性汇刊》，第 65 卷，第 2 期，第 1045-1057 页，doi：10.1109/TR.2015.2484064。

当前版本为版本 1.0。 稿件正在审稿。修改或扩展版本将发表于期刊。

29. Jang-wook Hur、Dhominick Pagulayan、Bo-Hyeon Kim、Kyeong-Hwan Hwang，《单次触发系统的可靠性建模与分析》，第2届工程与技术进展国际会议（RTET-2017）。

30. Moore，RE，1966，《区间分析》，美国普伦蒂斯-霍尔出版社。

31. Pereira，LES 与 Da Costa，VM，2014，《考虑负荷数据不确定性的电力系统最大负荷点区间分析》，《国际电力与能源系统期刊》，第54卷，第334-340页。

32. Kolev，LV，1993，《电路分析的区间方法》（第1卷），世界科学出版社。

33. Impollonia，N 与 Muscolino，G，2011，《具有不确定但有界轴向刚度的结构区间分析》，《计算机辅助力学与工程方法》，第200卷，第2期，第1945-1962页。

34. Wang, X, Yang, H, Wang, L & Qiu, Z 2012, '基于多种载荷工况的结构损伤识别区间分析方法'，Journal of Applied Mechanics，第79卷，第5期，第051010(1-8)页。

35. Wang, W, Xiong, J & Xie, M 2016, '参数不确定性条件下冷备用系统可靠性优化的区间分析研究'，Computers and Industrial Engineering，第97卷，第93-100页。

36. Bhunia, AK & Samanta, SS 2014, '区间度量研究及其在区间目标多目标优化中的应用'，Computers and Industrial Engineering，第74卷，第169-178页。

37. Gupta, RK, Bhunia, AK & Roy, D 2009, '一种基于遗传算法惩罚函数的求解方法，用于组件可靠性取值为区间的串联系统约束冗余分配问题'，Journal of Computational and Applied Mathematics，第232卷，第2期，第275-284页。

38. Mahato, SK & Bhunia, AK 2006，《面向区间运算的全局优化区间计算技术》，Applied Mathematics Research eXpress，2006 年第 2006 卷，第 1-19 页，https://doi.org/10.1155/AMRX/2006/69642

39. Sundaramoorthy, K, Purser, Z & Rotenberg, E 2000，'Slipstream 处理器：同时提升性能和容错能力'，ACM SIGPLAN Notices，第 35 卷，第 11 期，第 257-268 页。

40. LaFrieda, C, Ipek, E, Martinez, JF & Manohar, R 2007，'利用动态耦合内核构建具有韧性的芯片多处理器'，第三十七届 IEEE/IFIP 国际可靠系统与网络会议论文集，第 317-326 页。

41. Aggarwal, N, Ranganathan, P, Jouppi, NP & Smith, JE 2007，'可配置隔离：利用商用多核处理器构建高可用性系统'，ACM SIGARCH Computer Architecture News，第 35 卷，第 2 期，第 470-481 页。

42. Smolens, JC、Gold, BT、Falsafi,B和Hoe, JC 2006，'Reunion：复杂度高效的多核冗余'，第三十九届 IEEE/ACM 国际微体系结构研讨会论文集，第 223-234 页。

43.Kuo,W&Zuo, MJ2003, 最优可靠性建模：原理与应用。约翰·威立父子出版公司。

44. Roy, P, Mahapatra,BS,Mahapatra,GS & Roy,PK2014，'区间环境下基于熵的区域缩减遗传算法用于可靠性冗余分配'，《专家系统及其应用》，第 41 卷，第 14 期，第 6147-6160 页。

45. Shashikiran Venkatesha 和 Ranjani Parthasarathi，《基于单次系统的低成本容错计算系统可靠性建模与分析：由一个指令核组成》，2022

当前版本为版本 1.0。 稿件正在审稿。修改版或扩展版将在期刊上发表。

智能生成计算、通信与网络国际会议（SMART GENCON），印度班加罗尔，2022年，第1–9页，doi：10.1109/SMARTGENCON56628.2022.10084187。

46. Shashikiran Venkatesha 和 Ranjani Parthasarathi（2022）。《针对由单指令核组成的多核系统，使用基于区间的遗传算法和粒子群优化评估冗余分配问题》，载于：Arunachalam, V., Sivasankaran, K.（编）《微电子器件、电路与系统》。ICMDCS 2022。《计算机与信息科学通信》，第 1743 卷。施普林格，查姆。https://doi.org/10.1007/978-3-031-23973-1_4.

当前版本是版本 1.0。 稿件正在审稿中。修改或扩展后的版本将刊登于期刊。