DOI:10.7544/issn1000-1239.2016.20148204
53(3):682-696,2016

冗余及监控混合策略的优化配置算法研究

何 盼[1] 谭 春[1] 袁 月[1] 吴开贵[2]
[1](中国科学院重庆绿色智能技术研究院 重庆 400714)
[2](重庆大学计算机学院 重庆 400044)
(hepan@cigit.ac.cn)

Optimal Resources Allocation Algorithm for Optional Redundancy and Monitoring Strategies

He Pan[1], Tan Chun[1], Yuan Yue[1], and Wu Kaigui[2]
[1](*Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Science, Chongqing* 400714)
[2](*College of Computer Science, Chongqing University, Chongqing* 400044)

Abstract In big data environment, the use of optional redundancy and monitoring strategy in one system increases the usage of resource and causes state space expansion for optimal resources allocation model. The performance of existing evolutionary search algorithms should be improved for the solution space formed by both integer and non-integer variables. To improve the algorithm efficiency, a memetic algorithm based on triple element array is proposed on the analysis of search neighborhood. First of all, the impact of change of variables such as monitoring rate on the system reliability increase is analyzed and then changing-length neighbor generation method is proposed for monitoring rate on neighbor analysis. The neighbor generation method is also proposed for strategy options considering the relations between components. After that, local search operator is refined through the iterative search among components, which increases the search range while maintaining the local advantage of individuals. This operator is used for improving the whole framework of memetic algorithm. Experiment results indicates that this algorithm can be used to get the solution of strategy option of each component and the corresponding optimized parameters for multiple optional strategies. Compared with existing multi-strategy search algorithms, the improved memetic algorithm could get better resources allocation results under the same reliability constraint. The local search operator does not have great impact on the stability of the whole algorithm.

Key words redundancy allocation; monitoring resources allocation; reliability optimization; memetic algorithm; sensitivity analysis

摘 要 大数据环境中监控和冗余混合策略的采用引起资源优化配置模型的状态空间膨胀,进化搜索算法在整型与非整型变量结合的解空间中的搜索效率有待提高,为此提出了基于搜索邻域分析的三元组模因算法.在分析了监控频率等参数变化对组件及系统可靠性增长影响的基础上,针对监控频率提出了基于变长邻域的近邻生成方法,针对策略选项提出了与组件关联的近邻生成方法.采用模因算法框架并改进了局部搜索算子,通过组件间的迭代搜索在保持个体优势的同时增大搜索范围.该算法能够用于求解混合策略下的组件保障措施选项及相应优化配置参数;与现有多策略搜索算法相比,在相同可靠性约

收稿日期:2014-11-05;**修回日期**:2015-03-19
基金项目:国家自然科学基金项目(61309005);重庆市基础与前沿研究计划一般项目(cstc2014jcyjA40015)
This work was supported by the National Natural Science Foundation of China (61309005) and the Basic and Frontier Research Program of Chongqing (cstc2014jcyjA40015)
通信作者:袁月(yuanyue@cigit.ac.cn)

束下,该算法能够得到消耗更低的资源配置结果;局部搜索策略对算法稳定性未造成明显影响.

关键词　冗余度分配;监控资源分配;可靠性优化;模因算法;敏感度分析

中图法分类号　TP302.7; TP311.5

大数据环境中系统规模的增大和复杂性的提升为系统可靠性保障引入了新的研究焦点.大数据存储及处理系统一般持续运行时间较长,并且各个组件组成复杂,通常采用分布式系统架构进行数据运算[1],单种可靠性保障策略在其中具有一定局限性[2],系统可能同时出现多种可靠性策略的混合应用方式[3-4].但是任一可靠性保障策略的采用和多策略混合都会引起资源消耗增多,而大数据环境下的巨大能源消耗已成为亟待解决的问题[5],需要从适时节约资源的角度出发对每个组件中采用的不同策略及其相应参数进行优化选择和替换.虽然已有文献面向可靠性优化建立了混合多种保障策略如不同冗余机制[6-8]、软件重配置与周期再生[3]、冗余与非完美排错结合[4]的资源优化配置模型,但均未针对大数据环境下的可靠性保障策略进行分析.在分布式计算环境中,为不中断系统的正常持续执行,自适应策略如监控也是常用的系统可靠性维护机制[9].在保留一定积极冗余[10]资源的同时,基于监控的动态替换作为一种常用手段被用于系统可靠性的维护中[11].虽然现有研究提出了监控策略优化配置[12-13]及冗余与监控结合策略的优化模型[14],但在策略类型及参数的求解方面仍有一定不足.由于冗余或监控策略对系统可靠性影响的不同,资源优化配置通常为NP-难非线性优化问题,难以采用精确解法进行求解[15].而多保障策略的采用将增加优化模型中待求解变量的数量,使解变量的搜索空间呈指数级增长,增加进化算法搜索的难度.所以如何对冗余监控混合策略的变量空间进行深入分析并提高进化算法的搜索效率是本文主要需要解决的难题.

针对2种冗余机制混合的可靠性优化模型,Coit[16]采用了0-1整型规划方法求解每个组件所采取的冗余方式并得到相应的冗余度,Tavakkoli-Moghaddam等人[17]在三元组编码的基础上,采用改进遗传算法进行求解.以上文献主要针对单目标的冗余配置模型,Chambari等人[18]提出了面向多目标的多冗余策略配置优化模型,并采用了第2代快速非支配排序方法(NSGA Ⅱ)[19]进行个体选择.其后,Chambari等人[20]和Safari[21]分别采用了模拟退火算法以及基于Max-Min变异算子的遗传算法求解该多目标优化问题.以上2种算法中均采用了三元组编码方式,适用于搜索范围较大的整型参数变量,其搜索效果优于传统的遗传算法.以上2类研究虽然采用了不同的策略配置方式,但都局限于在冗余策略中进行选择,算法未对非整型变量的搜索作详细讨论.Nourelfath等人[4]考虑冗余与非完美排错结合的可靠性优化模型,采用划分空间的进化搜索算法搜索近优解,其中遗传算法被用于子空间的划分,禁忌搜索被用于子空间内部的局部搜索.在改进了非完美排错模型的基础上,Liu等人[22]采用遗传算法对冗余与非完美排错结合的优化模型进行求解.虽然以上算法适用于整型与非整型参数混合的变量求解,但对非整型数据的特征未做进一步分析,采用通用的搜索或选择算子,在复杂状态空间的搜索中仍存在一定问题.针对更为一般的软件系统,Levitin[23]采用遗传算法寻找不同软件版本的组合以及确定它们之间的执行顺序.Ahmadizar等人[24]采用了蚁群算法解决通用的优化配置模型,但仅用不同组件可靠性值及约束作为算法的输入,没有考虑具体保障策略的配置参数.

考虑大数据环境下冗余与监控策略并存的分布式系统,现有研究提供了基于三元组编码的模因算法进行求解[14],该算法的交叉变异及选择算子能够保证算法在搜索广度中的效果,但未对连续非整型变量的搜索进行深入讨论,在搜索深度方面还有待提高:

1) 对整型离散冗余变量和非整型连续监控频率变量采用相同结构或类似的搜索算子,虽然在离散的搜索域中能够获得较好效果,但在连续空间搜索域中将受到限制,容易陷入局部最优解.

2) 对不同参数变量如冗余变量和监控频率变量之间的内在制约关系未做深入分析,对单个组件资源配置参数与整体系统资源配置参数间的关系也未作分析,将不同组件、不同策略的变量视为相互独立的参数进行搜索,在解的搜索深度上将受到限制.

所以,本文提出了对混合的监控策略和冗余策略同时进行优化选择配置的改进模因算法,为保留进化算法搜索广度的要求,保留了已有算法中的交叉、变异以及选择算子,同时针对连续的监控频率变

量的特征,建立了变长邻域的迭代局部搜索算子用于增加搜索的深度.分析了积极冗余机制与基于监控的冗余替换机制的关系,在局部搜索过程中考虑了单个组件配置参数的变化对整体系统配置参数的影响,建立了改进的模因算法提高算法搜索效率和效果.本文首先分析了已有的冗余及监控混合策略的资源优化模型以及现有搜索算法的缺点;其次基于可靠性敏感度分析建立了监控频率局部搜索的变长邻域范围.在改进的局部搜索算子基础上建立了模因算法框架,并通过实验对比分析了改进算法与现有搜索算法的优缺点,验证了该算法在多策略优化配置问题上的优越性.

1 冗余及监控混合策略的资源优化模型

1.1 假设条件

对采用了冗余及监控混合可靠性保障策略的系统作4点假设:

1)假设未采用任何保障机制时,组件 i 及其冗余组件的有效工作时间 X_i 满足指数分布,即 $X_i \sim EXP(\lambda_i)$,其中 λ_i 表示未配置保障措施的单个组件 i 的失效率.

2)假设在同一时刻1个组件只能采取积极冗余或基于监控的冗余替换其中一种机制,不同的组件可以采用不同的机制.

3)为评估监控频率对组件可靠性的影响,假设基于监控的冗余替换机制中组件的冗余度固定为2,每个组件中设置监控的单元时间平均代价相同.

4)假设针对的系统均为分布式系统,即系统可靠性与组件可靠性间的关系可通过层次方法[12]计算获得.

1.2 融合2种保障策略的资源优化配置模型

针对可靠性保障策略进行资源优化的目的在于在有限的配置资源内达到系统可靠性的最大化或在保障系统可靠性的情况下进行资源的最小化配置.选用后者的优化思想建立融合冗余与监控2种保障策略的资源优化配置模型.

$$\begin{aligned} &\min\ C_{\text{sys}} = \sum_{i=1}^{m} x_i C_i;\\ &\min\ M_{\text{sys}} = \sum_{i=1}^{m} y_i \lambda_{m_i};\\ &\text{s.t. } R_{\text{sys}}(t) = \prod_{i=1}^{m} R_i(t)^{V_i} \geqslant R_0. \end{aligned} \tag{1}$$

在式(1)中,$R_i(t)$表示时刻 t 系统中组件 i 的可靠性,V_i 表示组件 i 的平均访问次数,m 表示系统中的组件个数.λ_{m_i} 表示组件 i 的监控频率,C_i 表示组件 i 的冗余代价.式(1)的约束条件为系统可靠性 $R_{\text{sys}}(t)$ 达到可靠性约束 R_0 的要求,优化目标为达到冗余配置资源 C_{sys} 与监控配置资源 M_{sys} 的最小化.x_i 与 y_i 分别为组件冗余代价及监控代价的系数.

当组件 i 采用单纯的积极冗余机制时,该组件的可靠性 $R_i(t)$ 如式(2)所示:

$$R_i(t) = 1-(1-\mathrm{e}^{-\lambda_i t})^{n_i}, \tag{2}$$

其中,λ_i 表示未配置冗余的单个组件 i 的失效率,n_i 表示组件的冗余度($1 \leqslant n_i \leqslant p$,$p$ 表示每个组件 i 的最大冗余度).此时冗余代价系数 $x_i = n_i$,监控代价系数 $y_i = 0$.

利用Markov链理论对组件可靠性及其与监控频率间的相关关系进行分析[25].当组件 i 采用基于监控的冗余替换机制时,该组件的可靠性 $R_i(t)$ 如式(3)所示[12]:

$$R_i(t) = -\lambda_i\left(\frac{4\lambda_i \mathrm{e}^{\frac{(-a_i+b_i)t}{2}}}{b_i(-a_i+b_i)} + \frac{4\lambda_i \mathrm{e}^{\frac{-(a_i+b_i)t}{2}}}{b_i(a_i+b_i)}\right), \tag{3}$$

其中,

$$a_i = 2\lambda_{m_i} + 3\lambda_i,$$

$$b_i = \sqrt{4\lambda_{m_i}^2 + 12\lambda_i\lambda_{m_i} + \lambda_i},$$

组件 i 的监控周期为 $1/\lambda_{m_i}$($0 \leqslant \lambda_{m_i} \leqslant q$,$q$ 表示每个组件 i 的最大监控频率).此时 $x_i = 2$,$y_i = 1$.

1.3 基于三元组编码的传统模因算法

对每个组件而言,式(1)中的待求参数包括组件策略选项、冗余度以及监控频率3个变量 $x_i = [n_i, \lambda_{m_i}, \alpha_i]$,其中 α_i 表示策略选择,故该优化问题是复杂的多元多目标非线性规划问题,难以采用传统线性规划的方法进行求解.为获取该问题的近似最优解,基于NSGA Ⅱ[19]的模因算法T-MOMA[14]被用于求解该问题.T-MOMA中采用了三元数组的编码方式对每个组件的策略选择方式、冗余度及监控频率进行编码.图1表示了6个组件组成的系统三元组编码,数组行1表示系统组件的冗余度,行2表示组件监控频率,行3代表可靠性保障策略选项(R:积极冗余,M:基于监控的冗余替换).

T-MOMA算法中的搜索策略存在2个弊端:

1)与传统进化搜索算法的解相比,三元编码方式的解空间范围增大,传统基于三元组编码的交叉、变异、局部搜索算子针对三元组中的某行或某列数据采取相同操作,在数据类型相同且搜索范围相近时(如针对不同类型的冗余度搜索),对不同参数可

		1	2	3	4	5	6
Component Redundancy	1	2	1	3	4	5	2
Monitoring Rate	2	0.002	0.01	0.045 8	0.266 8	0.99	0.1
Strategy Option	3	R	M	M	R	R	M

Fig. 1　Example of triple element encoding mechanism.

图 1 三元组编码示例

达到相似搜索效果.而监控频率参数为连续变量,监控频率参数的搜索空间远大于冗余参数的搜索范围.以 m 个组件的系统为例,冗余度范围在[1,5]内的整型变量,遍历搜索需要 5^m 次,而对监控频率在[0,1]之内的连续变量,建设搜索间隔为 0.01,则遍历搜索需要 100^m 次(减小搜索间隔还将增大搜索次数).此时对两者采用相同方法将很难对监控策略的解空间进行深入搜索,所以针对不同类型变量需要采用不同类型的搜索策略.

2) 现有算法将三元组数据中的每行数据作为孤立的变量进行搜索.而在优化搜索过程中保障策略选项的改变对组件以及系统可靠性的影响较大,在交叉、变异算子中可采用此方法,但在局部搜索中产生的新个体可能丧失局部优势,难以适应近邻产生的条件,降低了局部搜索的效果,此时需要同时对组件乃至系统中其他配置参数进行相应调整.

由于在模因算法中,局部搜索算子是增强算法搜索效率的重要因素[26],所以在第 2 节中主要针对局部搜索算子对以上 2 个问题进行改进.

2　基于邻域分析的局部搜索近邻生成算法

局部搜索算子中为了对选定个体的近邻进行搜索选择,首先需要确定近邻的选择方法以及范围.在现有 T-MOMA 算法中,对 $x_i=[n_i,\lambda_{m_i},\alpha_i]$ 中的 3 个变量分别在邻域范围内进行变化并产生相应的近邻.冗余度变量为离散变量,其近邻选择的邻域为[−1,1].监控频率为连续变量,其近邻选择的邻域范围为$[-q/2,q/2]$,近邻产生的改变值为$[-q/2,q/2]$中均匀分布的随机值,其中 q 为 λ_{m_i} 的取值范围,同式(3).虽然监控频率的近邻产生是通过随机变量实现,但其邻域取值固定,在监控频率取值较大时容易产生更优的近邻,但在取值较小时反而不能对较小范围内的 λ_{m_i} 值进行深入搜索.所以本节在可靠性参数敏感度分析的基础上,通过建立变长的邻域范围,改进局部搜索算子中的近邻生成方法.

2.1　基于敏感度分析的变长邻域范围确定

根据式(1)～(3)的可靠性分析模型,参数变量(包括冗余度或监控频率)的增加将引起组件及系统可靠性增长,但随着参数变量值的增大,可靠性的增长趋势将逐渐减弱.图 2 和图 3 分别表示了示例组件的可靠性随冗余度或监控频率增加的变化趋势.

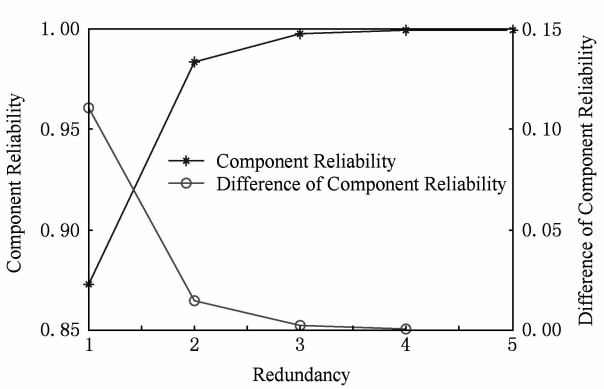
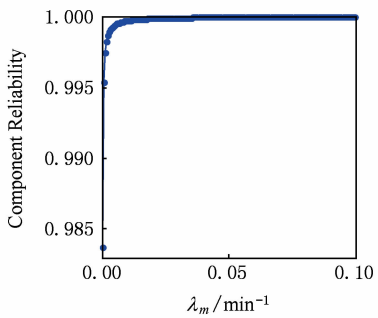
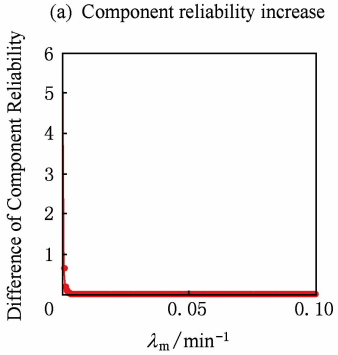


Fig. 2　Component reliability changing with the increase of redundancy.

图 2　组件可靠性随冗余度增加的变化趋势



(a) Component reliability increase



(b) Difference of component reliability decrease

Fig. 3　Component reliability changing with the increase of monitoring rate.

图 3　组件可靠性随监控频率增加的变化趋势

为对该影响关系作进一步分析,建立参数变量的变化对组件及系统可靠性增长的敏感度分析模型.

1) 参数变量对组件可靠性敏感度

冗余度及监控频率变化对组件的可靠性增长敏感度如式(4)~(5)所示,其中 a_i 与 b_i 同式(3).

$$\frac{\partial R_i(t)}{\partial n_i}=-(1-\mathrm{e}^{-\lambda_i t})^{n_i}\ln(1-\mathrm{e}^{-\lambda_i t}), \tag{4}$$

$$\frac{\partial R_i(t)}{\partial \lambda_{m_i}}=\frac{4\lambda_i}{(b_i)^{1.5}}((b_i t-2)\mathrm{e}^{\frac{-a_i+b_i}{2}t}+(b_i t+2)\mathrm{e}^{\frac{a_i+b_i}{2}t}). \tag{5}$$

2) 参数变量对系统可靠性敏感度

冗余度及监控频率变化对系统的可靠性增长敏感度如式(6)所示.

$$\begin{aligned}\frac{\partial R_{\mathrm{sys}}(t)}{\partial R_i(t)}&=\frac{V_i}{R_i(t)}\prod_{i=1}^{m}R_i(t)^{V_i},\\ \frac{\partial R_{\mathrm{sys}}(t)}{\partial n_i}&=\frac{\partial R_{\mathrm{sys}}(t)}{\partial R_i(t)}\times\frac{\partial R_i(t)}{\partial n_i},\\ \frac{\partial R_{\mathrm{sys}}(t)}{\partial \lambda_{m_i}}&=\frac{\partial R_{\mathrm{sys}}(t)}{\partial R_i(t)}\times\frac{\partial R_i(t)}{\partial \lambda_{m_i}}.\end{aligned} \tag{6}$$

根据组件敏感度分析的结果,虽然冗余度增长对可靠性变化的影响呈下降趋势,但其为固定范围内的离散变量,故对冗余度的局部搜索可选择[-1,1]的邻域范围. 与冗余度变化相比,监控频率的变化对可靠性增长的影响曲线较陡,同时,监控频率取值连续,难以对其影响进行评估. 如图 4 所示,当监控频率值较小(如 $\lambda_{m1}<0.005$)时,其细微的变化也将引起组件可靠性较大的增长,此时应选取范围较小的邻域进行近邻搜索;反之当监控频率值较大(如 $\lambda_{m2}>0.01$)时,频率值的变化对可靠性增长的影响较小,则此时的邻域范围需要相应增大.

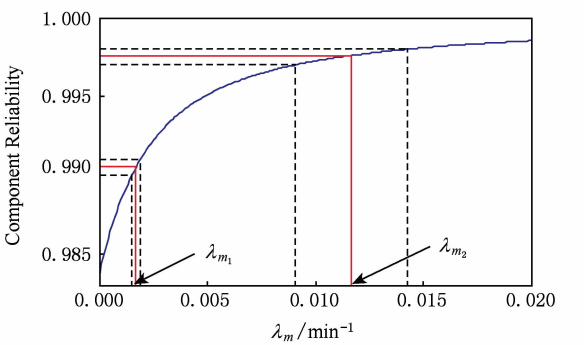


Fig. 4 The neighbor search range of different monitoring rate.

图 4 不同监控频率取值的邻域范围

采用固定长度的邻域或随机分布的邻域范围无法达到上述搜索目标. 所以基于敏感度分析,建立监控频率的局部搜索变化范围为$[-\Delta\lambda_{m_i},\Delta\lambda_{m_i}]$,如式(7)所示. 在选取固定 $\Delta R_{\mathrm{sys}}(t)$ 值的前提下,λ_{m_i} 取值越大,$[-\Delta\lambda_{m_i},\Delta\lambda_{m_i}]$ 范围越大,反之 λ_{m_i} 取值越小,邻域范围越小.

$$\Delta\lambda_{m_i}\approx\frac{\Delta R_{\mathrm{sys}}(t)}{\dfrac{V_i}{R_i(t)}R_{\mathrm{sys}}(t)\dfrac{\partial R_i(t)}{\partial\lambda_{m_i}}}. \tag{7}$$

2.2 基于邻域分析的近邻生成方法

对组件 $x_i=[n_i,\lambda_{m_i},\alpha_i]$ 中的 3 个变量分别进行不同变化可生成不同新个体用于局部算法搜索.

1) 基于冗余度变化的近邻生成

由于冗余度是固定范围[1,p]内的离散整数变量,对冗余度的改变仍然采用 T-MOMA 中加减 1 的方法. 新产生的个体 $x_i'=[n_i',\lambda_{m_i},\alpha_i]$,其中 $n_i'=n_i\pm1$. 该算法的伪代码如下:

算法 1. genNeighbour_N.

输入:需要进行局部搜索的个体 p_j、选定组件 i;

输出:更新后的个体 p_{j1},p_{j1}'.

① 生成新个体 $p_{j1}.n_i=p_j.n_i+1$,其他值与 p_j 相同;

② 生成新个体 $p_{j1}'.n_i=p_j.n_i-1$,其他值与 p_j 相同;

③ return p_{j1},p_{j1}'.

2) 基于监控频率变化的近邻生成

监控频率变化的邻域范围为$[-\Delta\lambda_{m_i},\Delta\lambda_{m_i}]$,对监控频率的改变采用邻域内的随机数进行修正. 假设新产生的个体 $x_i'=[n_i,\lambda_{m_i}',\alpha_i]$,$\lambda_{m_i}'=\lambda_{m_i}\pm\varepsilon_i$,$\varepsilon_i$ 值如式(8)所示,U_i 为服从范围$[-\Delta\lambda_{m_i},\Delta\lambda_{m_i}]$均匀分布的随机变量,$y$ 为近邻搜索的循环次数.

$$\begin{aligned}\varepsilon_i&=\frac{U_i}{2^{y-1}},\\ U_i&\sim U[-\Delta\lambda_{m_i},\Delta\lambda_{m_i}].\end{aligned} \tag{8}$$

算法 2. genNeighbour_Lambda.

输入:需要进行局部搜索的个体 p_j、选定组件 i;

输出:更新后的个体 p_{j2},p_{j2}'.

① 根据式(7)以及 $p_j.\lambda_{m_i}$,计算相应 $\Delta\lambda_{m_i}$;

② 定义近邻搜索最大循环次数 z;

③ for $y=1:z$

④ 利用式(8)、y 值以及第①行获得的 $\Delta\lambda_{m_i}$ 计算 ε_i;

⑤ 生成新个体 $p_{j2}.\lambda_{m_i}=p_j.\lambda_{m_i}+\varepsilon_i$,其他值与 p_j 相同;

⑥ if p_{j2} 达到可靠性约束条件

⑦ 停止计算 p_{j2};

⑧ end if

⑨　生成新个体 $p'_{j2}.\lambda_{m_i}=p_j.\lambda_{m_i}-\varepsilon_i$,其他值与 p_j 相同;

⑩　if p'_{j2}达到可靠性约束条件

⑪　　停止计算 p'_{j2};

⑫　end if

⑬　if p_{j2}与 p'_{j2}均停止计算

⑭　　break;

⑮　end if

⑯ end for

⑰ return p_{j2},p'_{j2}.

3) 基于策略选项混合变化的近邻生成

策略选项 α_i 为二元变量,其近邻的产生只能通过改变 α_i 值获得. 与前 2 种生成方法不同的是,相反策略选项的选择可能引起相应的冗余度或监控频率较大的变化,影响待搜索个体的局部优势,需要对三元组的其他参数变量做相应调整. 所以除了简单策略选项变化的近邻生成方法外,还可通过同时改变冗余度或监控频率形成新的近邻. 如对 $x_i=[n_i,\lambda_{m_i},\mathrm{M}]$可以产生近邻 $x'_i=[n_i\pm1,\lambda_{m_i},\mathrm{R}]$,如图 5 所示;对 $x_i=[n_i,\lambda_{m_i},\mathrm{R}]$可生成近邻 $x'_i=[n_i,\lambda_{m_i}\pm\varepsilon_i,\mathrm{M}]$. 在某些特定的情况下,除对本组件的其他参数变量进行优化外,还需要对个体中其他组件的参数变量进行变异. 例如在某组件三元组变量从 $x_i=[2,\lambda_{m_i},\mathrm{R}]$向 $x'_i=[2,\lambda_{m_i},\mathrm{M}]$的转化过程中,策略选项的改变不影响系统冗余度的变化(因为监控替换方法仍需要组件冗余度为 2),但将显著增加组件监控频率,对可靠性变化影响也较大,此时需对系统中其他采用了监控策略的组件中监控频率做相应优化如:$\lambda'_{m_k}=\lambda_{m_k}-\varepsilon_k$,$1\leqslant k\leqslant m$ & $\alpha_k=\mathrm{M}$,如图 6 所示. 该算法的伪代码可参见算法 3.

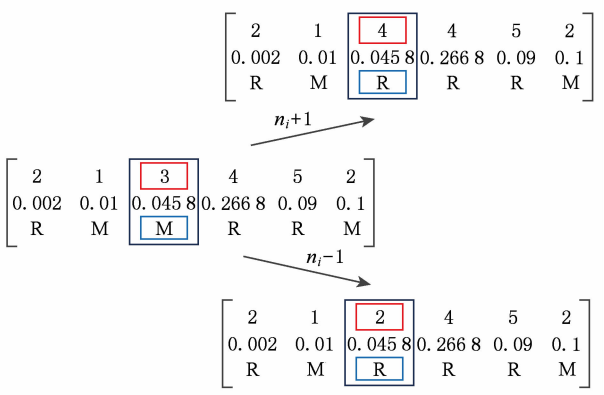


Fig. 5　Neighbor generation based on the redundancy and strategy change.

图 5　基于冗余度及策略选项混合变化的近邻生成方法

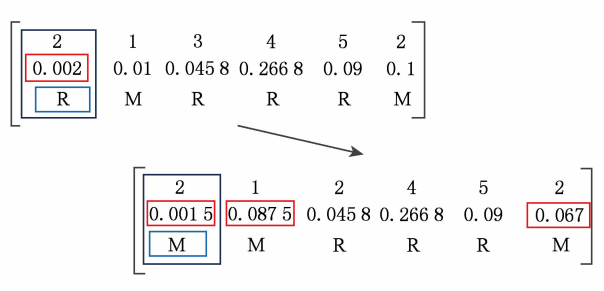


Fig. 6　Neighbor generation based on the monitoring rate and strategy change.

图 6　基于监控频率及策略选项混合变化的近邻生成方法

算法 3. genNeighbour_Strategy.

输入:需要进行局部搜索的个体 p_j、选定组件 i;

输出:更新后的个体 p_{j3},p_{j4},p'_{j4},p_{j5},p'_{j5},p_{j6}.

① 生成新个体 $p_{j3}.\alpha_i=p_j.\alpha_i$ 的反值,其他值与 p_j 相同;

② if $p_{j3}.\alpha_i==\mathrm{R}$

③　$[p_{j4},p'_{j4}]=genNeighbour\_N(p_{j3},i)$;

④ else

⑤　$[p_{j5},p'_{j5}]=genNeighbour\_Lambda(p_{j3},i)$;

⑥　if $p_{j3}.n_i==2$

⑦　　生成新个体 $p_{j6}=p_{j3}$;

⑧　　for $k=1:m$

⑨　　if $p_{j3}.\alpha_k==\mathrm{M}$

⑩　　根据式(8)、式(9)以及 $p_{j3}.\lambda_{m_i}$ 计算相应 ε_k;

⑪　　$p_{j6}.\lambda_{m_k}=p_{j3}.\lambda_{m_k}-\varepsilon_k$;

⑫　　end if

⑬　end for

⑭　end if

⑮ end if

⑯ return p_{j3},p_{j4},p'_{j4},p_{j5},p'_{j5},p_{j6}.

3　基于迭代局部搜索策略的改进模因算法

在已有的局部搜索算子中,文献[21]采用了 Max-Min 变异方法选定个体中具有最大可靠性和最小可靠性的三元组进行变异,将某一组件的参数变量视为整体进行操作,在保证可靠性接近约束条件的同时可能减少资源消耗,但算法针对特定的三元组进行变化可能减低个体的部分优势. 文献[14]针对 3 种变量分别进行迭代搜索,即对不同组件的同一变量采取统一操作. 该方法能够提高搜索的范围,但搜索过程脱离了组件自身的可靠性分析,容易进入局部最优解. 本节中结合以上 2 种方法各自的

优点建立改进的迭代局部搜索算子并用以改进模因算法.

3.1 迭代局部搜索算法

针对选定的个体 p_j 的近邻,局部搜索算法迭代的搜索该个体的近邻,在每一次迭代中,随机选择个体中的任意组件 i,利用第 2 节中的近邻生成方法对其三元组数据 $x_i=[n_i,\lambda_{m_i},\alpha_i]$进行变化产生近邻.该方法每次主要对个体中的 1 个组件进行特定改变,能够保持该个体的优势特征;同时通过多次迭代过程增大局部搜索的范围.每次迭代产生近邻个体的详细步骤如图 7 所示:

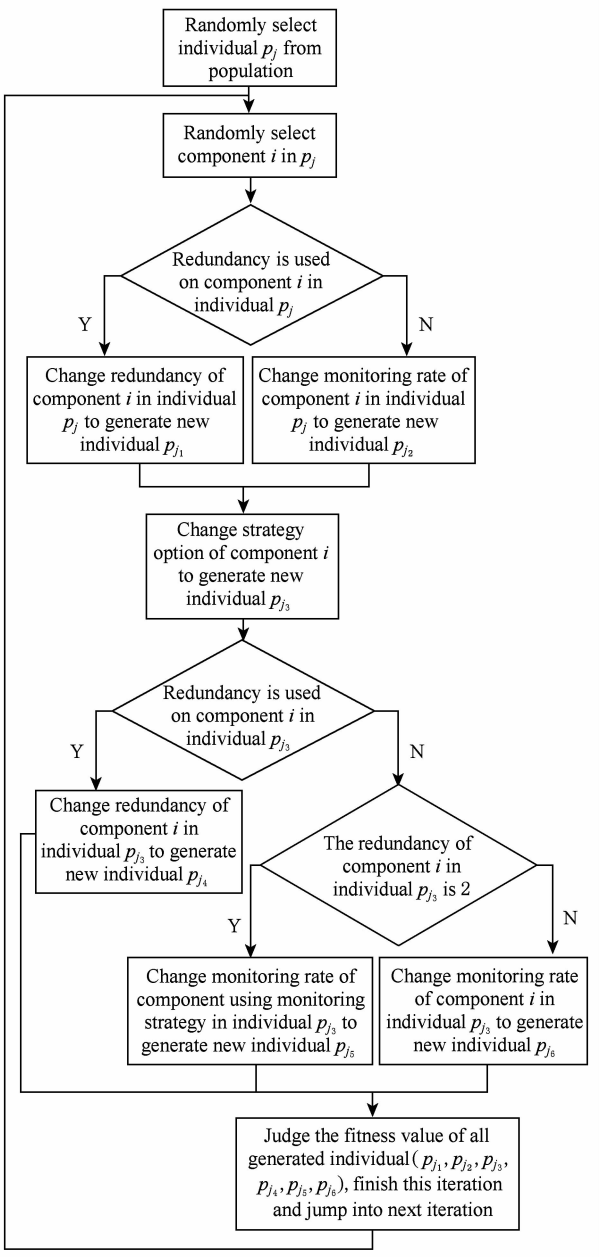


Fig. 7 Iterative search strategy in local search algorithm.

图 7 局部搜索算法中迭代搜索策略

在迭代搜索每次近邻产生的同时检查其约束值以及目标值与原个体之间的支配关系,能够支配原始个体或不被原始个体支配的新个体将被加入子种群,继续下一次交叉变异计算.局部搜索算法的完整框架所示如下:

算法 4. 迭代局部搜索算法 localsearch.

输入:需要进行局部搜索的个体 p_j、可行解集 Q、算法迭代次数 l、最大迭代次数 $l_{\max}$;

输出:更新后的可行解集 Q.

① if $l\leqslant l_{\max}$,执行后续搜索;
② 在个体 p_j 中随机选择一个组件 i;
③ if $p_j.\alpha_i==R$
④ $[p_{j1},p'_{j1}]=genNeighbour\_N(p_j,i)$;
⑤ if p_{j1} 与 p_j 互为非支配解,或 p_{j1} 支配 p_j
⑥ 将 p_{j1} 加入可行解 Q 中;
⑦ end if
⑧ $localsearch(p_{j1},Q,l+1,l_{\max})$;
⑨ 同理检查 p'_{j1} 是否加入可行解 Q 中;
⑩ $localsearch(p'_{j1},Q,l+1,l_{\max})$;
⑪ else
⑫ $[p_{j2},p'_{j2}]=genNeighbour\_Lambda(p_j,i)$;
⑬ 同理检查 p_{j2} 与 p'_{j2} 是否加入可行解 Q 中;
⑭ $localsearch(p_{j2},Q,l+1,l_{\max})$;
⑮ $localsearch(p'_{j2},Q,l+1,l_{\max})$;
⑯ end if
⑰ $[p_{j3},p_{j4},p'_{j4},p_{j5},p'_{j5},p_{j6}]=genNeighbour\_Strategy(p_j,i)$;
⑱ $localsearch(p_{j3},Q,l+1,l_{\max})$;
⑲ if p_{j3} 与 p_j 互为非支配解,或 p_{j3} 支配 p_j
⑳ 将 p_{j3} 加入可行解 Q 中;
㉑ else
㉒ if $p_{j3}.\alpha_i==R$
㉓ 检查 p_{j4} 与 p'_{j4} 是否加入可行解 Q 中;
㉔ $localsearch(p_{j4},Q,l+1,l_{\max})$;
㉕ $localsearch(p'_{j4},Q,l+1,l_{\max})$;
㉖ else
㉗ 检查 p_{j5} 与 p'_{j5} 是否加入可行解 Q 中;
㉘ $localsearch(p_{j5},Q,l+1,l_{\max})$;
㉙ $localsearch(p'_{j5},Q,l+1,l_{\max})$;
㉚ if $p_{j3}.n_i==2$
㉛ 检查 p_{j6} 是否加入可行解 Q 中;
㉜ $localsearch(p_{j6},Q,l+1,l_{\max})$;
㉝ end if

㉞　　end if

㉟　end if

㊱ end if

㊲ return Q.

3.2　改进模因算法框架

基于3.1节的迭代局部搜索算法4,改进模因算法(UD-TMOMA)框架所示如下:

算法5. UD-TMOMA.

输入:父种群 $P_0=\varnothing$、子种群 $Q=\varnothing$、进化代数 $t=0$、最大进化代数 $t_{\max}$;

输出:结果子种群 Q、进化代数 t.

① 种群初始化:$P_0=(p_1,p_2,\cdots,p_N)$,其中 p_1,$p_2,\cdots,p_N$ 为随机生成的个体;

② while $t=0:t_{\max}$

③　对P_t 执行以下操作:基于三元组编码的交叉以及变异操作产生子种群 Q;

④　for $j=1:N\times$局部搜索概率

⑤　　从子种群 Q 中随机选择1个个体 p_j;

⑥　　$localsearch(p_j,Q,0,l_{\max})$得到局部优化后的子种群 Q;

⑦　end for

⑧　合并父种群 P_t 和子种群 Q 生成新种群 R;

⑨　计算种群 R 中各个体的约束值和目标值;

⑩　采用第2代快速非支配排序方法,按照非支配字体和聚集距离对 R 进行倒序排序;

⑪　选择 R 中前 N 个个体,并放入下一代父种群 P_{t+1} 中;

⑫　$t=t+1$;

⑬ end while

算法中的种群初始化、交叉变异及排序操作与T-MOMA算法类似,可参见文献[14],在本文中不作详细阐述.

3.3　算法时间复杂度分析

与T-MOMA算法或传统基于Max-Min算子的遗传算法(GA)相比,UD-TMOMA算法的其他操作与其相同,时间复杂度的主要差别体现在局部搜索算子中.假设每次局部搜索迭代次数为 n,根据个体的情况,UD-TMOMA算法的局部搜索算子每次可能产生3~6个新个体,所以时间复杂度在 $O(3^n)$到 $O(6^n)$之间,传统T-MOMA算法对每行向量分别采用迭代搜索,每行向量每次产生2个新个体,时间复杂度为 $O(3\times 2^n)$,而GA算法每次仅对2个特定组件进行操作,时间复杂度为 $O(1)$.所以UD-TMOMA算法的局部搜索算子时间复杂度有一定增加,在迭代过程中迭代次数的选择对算法时间增长有较显著效果.

4　实验过程与结果分析

4.1　实验设置

本节我们采用文献[27]中的系统结构和相应参数进行实验.系统的组织结构如图8所示:

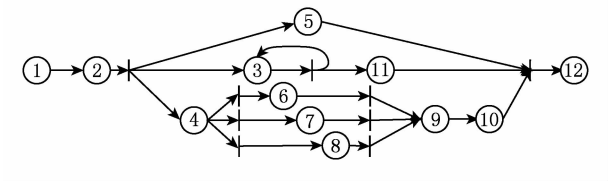


Fig. 8　Example of system architecture.

图8　系统结构示例

系统中采用的示例实验参数如表1所示:

Table 1　The Input Parameters of Units

表1　系统的输入参数值

Component No.	λ/min^{-1}	C
1	0.000 227 14	2
2	0.000 271 92	3
3	0.000 445 14	2
4	0.000 298 17	4
5	0.000 678 8	5
6	0.000 361 96	3
7	0.000 454 51	2
8	0.000 473 22	3
9	0.000 669	4
10	0.000 181 54	5
11	0.000 679 23	2
12	0.000 485 55	3

图8所示的系统被抽象为2部分:组件及组件间的状态转移关系.采用基于体系结构的可靠性分析方法[28],原始状态下的系统可靠性 $R_{sys}\approx R_1R_2R_3^{3.03}R_4R_5R_6^{0.7}R_7^{0.2}R_8^{0.1}R_9R_{10}R_{11}R_{12}$.系统中

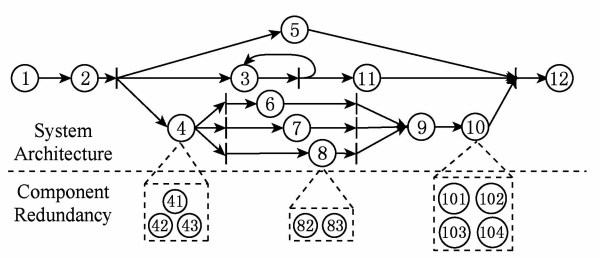


Fig. 9　Example of component redundancy.

图9　系统冗余示例

的单个组件提供了特定的功能,为了保障系统可靠性,在实际运行过程中,可能存在多个具有相同功能的组件,其中,1 个为正在使用的组件而其他组件均作为它的冗余备份存在,如图 9 所示.

每个组件中均可采用积极冗余[10]以及基于监控的冗余替换[12]2 类机制进行冗余组件的替换. 当组件采用积极冗余替换机制时,可替换的组件均处于运行状态中,组件可靠性主要取决于可替换的备份数量. 采用监控替换机制时,被监测到故障的组件可以得到及时的替换,组件的可靠性主要依赖于监控机制的频率. 考虑到 2 类机制共存于系统组件中,则需要同时对组件的冗余度以及监控频率进行计算,建立该系统的优化模型同式(1),故将该系统模型用于本文中的 UD-TMOMA 算法实验.

经过多次实验,UD-TMOMA 算法的参数设置如下:在邻域范围确定中,$\Delta R_{sys}(t)=0.001$,种群个位为 100,循环次数 $t_{max}=500$,交叉概率为 0.9,变异概率为 0.1,局部搜索概率为 0.05,为减少局部搜索引起的时间消耗,局部搜索迭代次数 $l_{max}=3$,局部搜索中监控频率搜索的最大循环次数 $z=3$.

4.2 案例分析

为对冗余及监控混合策略的优化配置结果做详细说明,在本实验中采用 UD-TMOMA 对特定条件下的系统进行运算,得到相应的资源配置结果如表 2 所示. 其中,t 表示选定的系统时间,R_0 表示相应的可靠性约束条件. 由于 UD-TMOMA 算法的计算结果是 1 组平衡解集,表 2 中针对每组实验数据仅选择了其中具有代表性的 2 组结果.

Table 2 An Case Study of Example System

表 2 系统案例结果示例

Component No.	$t=10$ h, $R_0=0.90$				$t=30$ h, $R_0=0.90$				$t=10$ h, $R_0=0.99$			
	min C_{sys}		min M_{sys}		min C_{sys}		min M_{sys}		min C_{sys}		min M_{sys}	
	n	λ_m/min^{-1}	n	λ_m/min^{-1}	n	λ_m/min^{-1}	n	λ_m/min^{-1}	n	λ_m/min^{-1}	n	λ_m/min^{-1}
1	2	0.0019	3	0.0000	2	0.0882	5	0.0000	4	0.0000	5	0.0000
2	2	0.0057	3	0.0000	2	0.1157	5	0.0000	2	0.0896	5	0.0000
3	2	0.0293	5	0.0000	2	0.3074	2	0.0648	2	0.2028	5	0.0000
4	2	0.0198	3	0.0000	2	0.1084	2	0.0240	2	0.1094	5	0.0000
5	2	0.0306	3	0.0000	2	0.2913	2	0.0572	2	0.2145	2	0.1298
6	2	0.0089	3	0.0000	2	0.1669	2	0.0242	2	0.0836	5	0.0000
7	2	0.0050	2	0.0000	2	0.0935	2	0.0164	2	0.0623	5	0.0000
8	1	0.0000	2	0.0000	1	0.0000	2	0.0129	2	0.0442	5	0.0000
9	2	0.0177	4	0.0000	2	0.2910	2	0.0524	2	0.1954	2	0.1409
10	2	0.0064	2	0.0000	2	0.0817	5	0.0000	2	0.0545	4	0.0000
11	2	0.0306	5	0.0000	2	0.2945	2	0.0566	2	0.1568	2	0.1297
12	2	0.0284	4	0.0000	2	0.1983	2	0.0396	2	0.1440	5	0.0000
C_{sys}	73		119		73		106		80		152	
M_{sys}/min^{-1}	0.1844		0		2.0370		0.3481		1.3570		0.4003	

表 2 显示在不同约束条件下,该算法均可用于同时计算每个组件的可靠性保障措施选项以及相应的保障策略参数(如冗余度或监控频率). 当时间增加或系统可靠性需求增长时,需要消耗更多的资源配置,不同组件冗余度或监控频率之间的增长相互影响,该特性与 T-MOMA 算法体现的结果一致.

4.3 算法对比分析

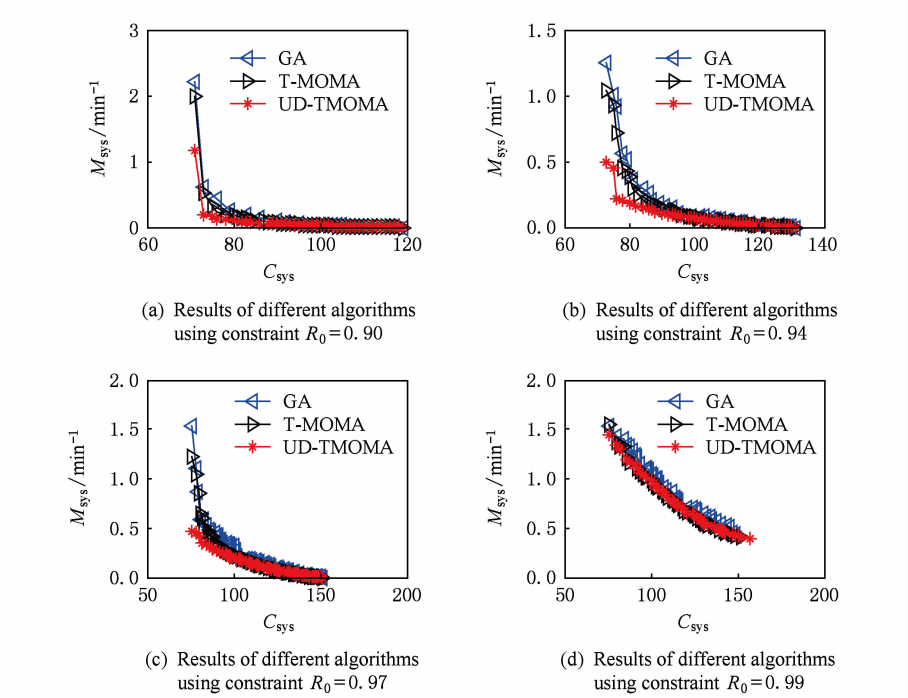
为了证明算法在局部搜索中的优势,在本节中,将 UD-TMOMA 算法与已有文献中采用了三元组编码的 GA[21]和 T-MOMA[14]算法比较. 目前基于三元组的 GA 算法是在多选择策略的系统冗余度分配领域的通用算法. 为了增强算法的搜索效率,GA 算法中采用了 Max-Min 变异算子,T-MOMA 算法则采用了其他局部搜索策略.

1) 不同可靠性约束

在不同的可靠性约束条件下,采用 3 种算法分别

计算近似最有资源优化配置模型,计算获得的 2 种资源消耗的解集如图 10 所示. 其中假设 $t=10$ h,选用了表 1 所示的系统输入值,设置可靠性约束从 0.90 变化至 0.99.

图 10 显示 UD-TMOMA 算法获得配置方法所消耗的资源均低于其他 2 种方法计算获得的资源.



(a) Results of different algorithms using constraint $R_0=0.90$

(b) Results of different algorithms using constraint $R_0=0.94$

(c) Results of different algorithms using constraint $R_0=0.97$

(d) Results of different algorithms using constraint $R_0=0.99$

Fig. 10　Results of three algorithms using different reliability constraints.

图 10　3 种算法对不同可靠性约束的实验结果

为了对算法的效果进行定量比较,采用了覆盖率[29]作为度量标准,覆盖率体现了 2 种解集相互支配的程度. 由于以上算法均为近似算法,具有一定随机性,所以对每种算法运行 10 次,所得到的每组优化配置结果对应的 2 种系统代价值的覆盖率如表 3 所示.

Table 3　The Fraction of Nondominated Solutions Covered by Other Nondominated Points Using Different Constraints

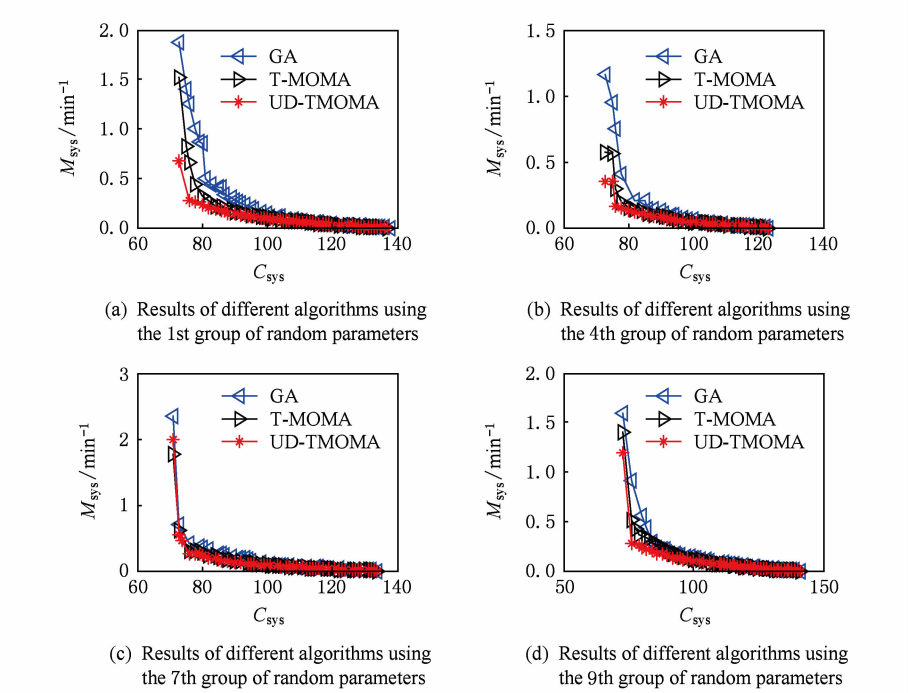
表 3　3 种算法在不同可靠性约束下的覆盖率

Reliability Constraint	GA/T-MOMA	T-MOMA/GA	GA/UD-TMOMA	UD-TMOMA/GA	T-MOMA/UD-TMOMA	UD-TMOMA/T-MOMA
0.90	0.4364	0.9914	0.0000	**0.9992**	0.0010	**0.9992**
0.91	0.3647	0.9850	0.0012	**0.9996**	0.0006	**0.9996**
0.92	0.5215	0.6251	0.0000	**1.0000**	0.0005	**0.9987**
0.93	0.1362	0.8933	0.0000	**0.9991**	0.0000	**1.0000**
0.94	0.3522	0.8957	0.0000	**0.9991**	0.0000	**0.9995**
0.95	0.1209	0.8096	0.0000	**0.9984**	0.0000	**0.9995**
0.96	0.0909	0.7103	0.0000	**1.0000**	0.0000	**0.9985**
0.97	0.2417	0.4492	0.0000	**0.9994**	0.0000	**1.0000**
0.98	0.2998	0.5733	0.0000	**0.9956**	0.0051	**0.9921**
0.99	0.1200	0.8305	0.0000	**0.9318**	0.0000	**0.8864**
Average	0.2684	0.7763	0.0001	**0.9922**	0.0007	**0.9874**

Note: A/B stands for A covers B.

2) 不同系统参数

除了不同的约束条件外,在 $t=10$ h 时也采用不同的系统组件失效率对 3 种算法进行实验,实验结果所消耗的资源如图 11 所示,该实验中设定可靠性约束为 0.95,选取 10 组随机的系统组件失效率值进行实验. 表 4 显示了不同测试用例下相应的覆盖率.



(a) Results of different algorithms using the 1st group of random parameters

(b) Results of different algorithms using the 4th group of random parameters

(c) Results of different algorithms using the 7th group of random parameters

(d) Results of different algorithms using the 9th group of random parameters

Fig. 11 Results of three algorithms using different system parameters.

图 11 3 种算法对不同系统参数的实验结果

Table 4 The Fraction of Nondominated Solutions Covered by Other Nondominated Points in Different Test Cases

表 4 3 种算法在不同测试用例下的覆盖率

Test Case	GA/T-MOMA	T-MOMA/GA	GA/UD-TMOMA	UD-TMOMA/GA	T-MOMA/UD-TMOMA	UD-TMOMA/T-MOMA
1	0.060 6	0.766 1	0.001 7	**0.997 5**	0.000 0	**1.000 0**
2	0.102 8	0.985 3	0.000 0	**1.000 0**	0.000 0	**1.000 0**
3	0.362 0	0.995 4	0.001 5	**0.999 3**	0.000 0	**1.000 0**
4	0.186 0	0.983 4	0.000 0	**0.999 2**	0.000 0	**1.000 0**
5	0.524 0	0.934 9	0.000 0	**1.000 0**	0.000 0	**1.000 0**
6	0.255 7	0.975 9	0.000 0	**0.999 5**	0.000 0	**1.000 0**
7	0.260 1	0.906 9	0.000 0	**0.999 6**	0.000 0	**1.000 0**
8	0.350 8	0.895 4	0.001 6	**0.996 5**	0.000 0	**0.999 5**
9	0.322 1	0.896 0	0.000 0	**0.999 6**	0.000 0	**1.000 0**
10	0.099 1	0.888 3	0.000 0	**0.998 9**	0.000 0	**1.000 0**
Average	0.252 3	0.922 8	0.000 5	**0.999 0**	0.000 0	**1.000 0**

Note: A/B stands for A covers B.

从解集覆盖率的结果可以看到 GA 算法与 T-MOMA 算法的结果基本都被新算法 UD-TMOMA 所支配,在可靠性约束较低即解空间较大时尤为明显. 为体现不同算法的解集在解空间的覆盖程度,以上解集计算的超体积指标(hypervolume indicator, HI)[30]值如表 5 所示. HI 值体现了解集在空间的分布程度.

从 HI 值可以看出 3 种算法在解空间的分布情况基本相同,其中 UD-TMOMA 算法的分布结果比 T-MOMA 算法略差.

3) 与单策略算法比较

在本节中,基于混合多策略的资源优化配置方法与已有文献中基于单策略的可靠性优化方法包括积极冗余配置方法[31]以及监控替换配置方法[12]进

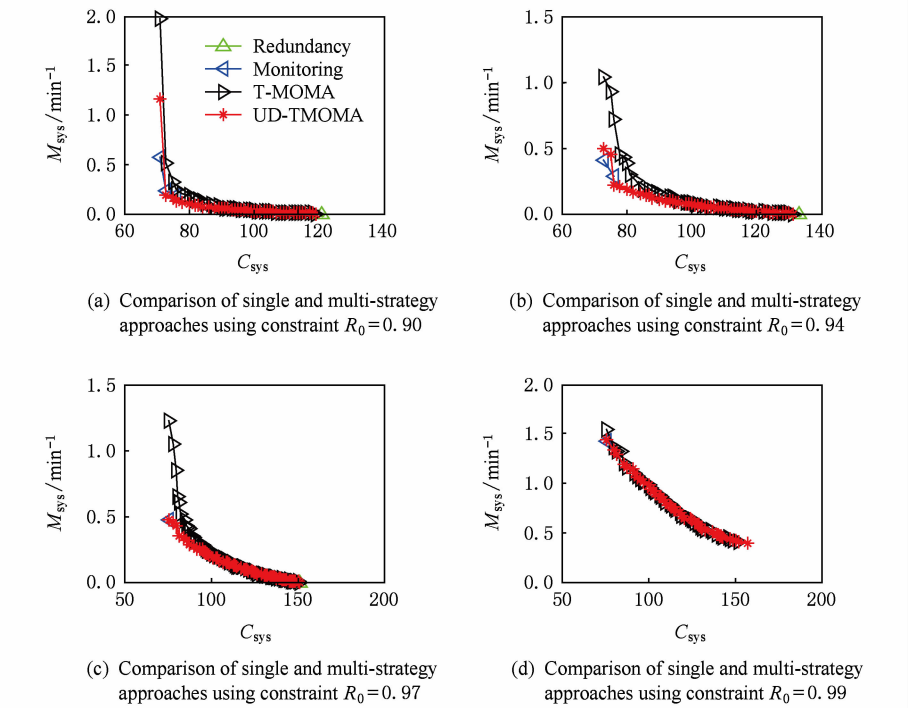
Table 5　Hypervolume Indicator of Different Algorithms under Different Conditions

表 5　UD-TMOMA 与其他算法在不同条件下的 HI 值

Reliability Constraint	Hypervolume Indicator			Test Case	Hypervolume Indicator		
	GA	T- MOMA	UD- TMOMA		GA	T- MOMA	UD- TMOMA
0.90	0.340 0	0.366 9	0.370 0	1	0.344 7	0.360 2	0.384 7
0.91	0.341 4	0.380 0	0.387 1	2	0.355 2	0.372 8	0.317 2
0.92	0.327 0	0.375 3	0.298 7	3	0.325 0	0.339 7	0.272 0
0.93	0.345 3	0.378 7	0.319 2	4	0.335 9	0.332 3	0.330 3
0.94	0.365 3	0.385 6	0.356 9	5	0.345 3	0.348 2	0.320 2
0.95	0.378 5	0.411 7	0.389 5	6	0.321 3	0.328 6	0.387 4
0.96	0.403 8	0.435 5	0.423 4	7	0.349 4	0.364 6	0.287 1
0.97	0.405 1	0.427 7	0.327 1	8	0.359 7	0.359 5	0.432 8
0.98	0.308 3	0.375 0	0.339 5	9	0.324 1	0.329 2	0.330 9
0.99	0.268 7	0.312 0	0.229 5	10	0.347 7	0.362 6	0.422 1
Average	0.348 3	0.384 8	0.344 1		0.340 8	0.349 8	0.348 5

行了对比. $t=10$ h 时同样设置可靠性约束从 0.90 变化至 0.99,在各种参数环境下分别运行基于 UD-TMOMA 的多策略方法以及基于 MOMA[12] 的监控资源分配和基于线性规划的冗余资源优化方法[31]. 不同方法进行可靠性优化后的资源消耗结果集如图 12 所示,为了更好地进行对比分析,T-MOMA 算法的结果也在图 12 中有所体现. 然后设定可靠性约束为 0.95,选取 10 组随机的系统组件失效率值,得到的相应资源消耗结果如图 13 所示.

上述实验结果中可以看到在可靠性约束较低的情况下(如 $R_0=0.91,0.94$),单种监控策略在同等监控频率下消耗的冗余资源略少于混合策略计算得到的结果,对于 T-MOMA 算法尤其明显. 采用 UD-TMOMA 算法,该情况略有改善.



(a) Comparison of single and multi-strategy approaches using constraint $R_0=0.90$

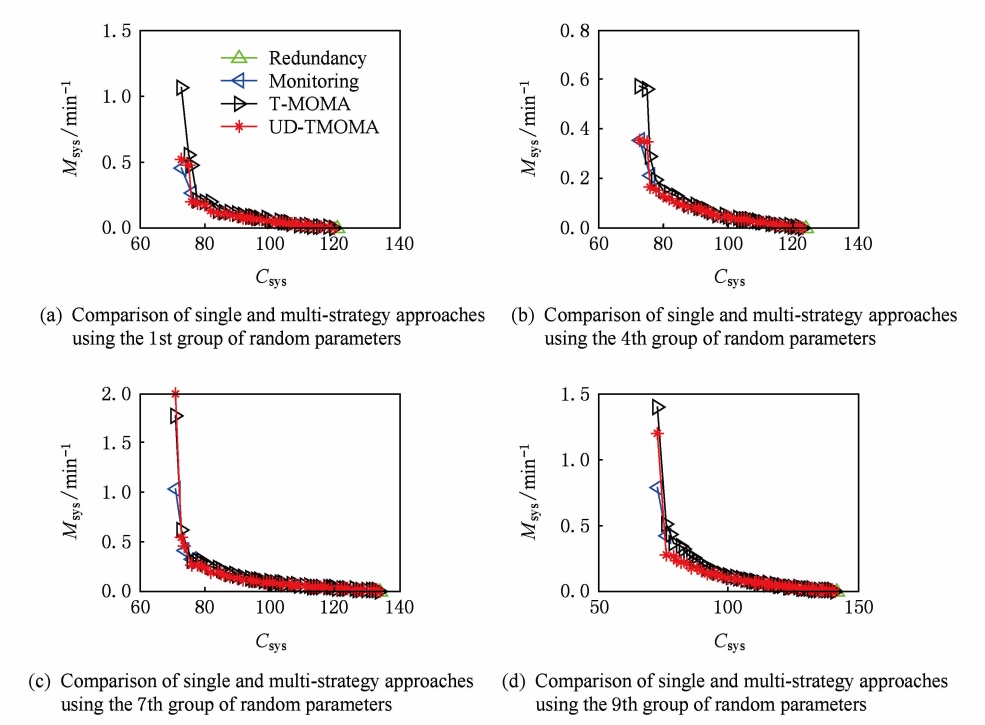
(b) Comparison of single and multi-strategy approaches using constraint $R_0=0.94$

(c) Comparison of single and multi-strategy approaches using constraint $R_0=0.97$

(d) Comparison of single and multi-strategy approaches using constraint $R_0=0.99$

Fig. 12　Resources comparison under different availability constraint.

图 12　不同可靠性约束下的资源消耗对比



(a) Comparison of single and multi-strategy approaches using the 1st group of random parameters

(b) Comparison of single and multi-strategy approaches using the 4th group of random parameters

(c) Comparison of single and multi-strategy approaches using the 7th group of random parameters

(d) Comparison of single and multi-strategy approaches using the 9th group of random parameters

Fig. 13 Resources comparison using different system parameters.

图 13 不同系统参数下的资源消耗对比

4.4 实验结果分析

从 4.3 节实验中我们可以得出 2 个结论:

1) 虽然 UD-TMOMA 算法中局部搜索策略的改变对解集在解空间的分布情况稍有降低,但没有对算法的稳定性造成明显影响,但在搜索性能方面有明显增加.

2) 由于单策略的搜索算法的参数变量较少,解空间小于混合策略的搜索算法,所以当可靠性约束较低时,单策略的搜索算法比混合策略的搜索算法获得的解略好,随着解空间减小,这种优势逐渐减弱. 在与单策略搜索算法的比较中,UD-TMOMA 算法的结果仍然优于 T-MOMA 算法.

总的来说,与其他多策略资源搜索算法相比,在不影响算法稳定性的前提下,UD-TMOMA 能够获得更优的资源配置结果. 在搜索空间较大的情况下,与单策略资源搜索算法相比还略有不足.

5 结束语

本文中我们主要研究了面向可靠性优化设计的冗余和监控混合可靠性保障策略的资源优化配置算法. 针对现有进化搜索算法在多元多维解空间中搜索力度的不足,建立了基于邻域分析的改进模因算法. 首先对冗余度及监控频率等参数变量变化对组件及系统可靠性增长影响进行了分析. 基于敏感度分析的结果建立了针对 3 种参数变量的近邻生成方法,例如针对监控频率搜索提出了基于变长邻域的近邻生成方法,针对策略选项搜索提出了与组件关联的系统近邻生成方法. 采用改进近邻生成方法建立了局部搜索算子,通过组件间的迭代搜索在保持个体优势的同时增大搜索范围,并用以改进模因算法框架. 通过实验验证了:该算法能够用于求解每个组件保障措施的选择以及相应的配置参数;与现有 2 种多策略搜索算法相比,改进模因算法能够获得取得更优的资源配置结果,在待求解空间较大时,优化效果尤为明显;虽然局部搜索策略使解集在解空间的分布略有降低,但对算法稳定性未造成明显影响. 在本文中监控策略与冗余策略作为独立的措施被应用于系统组件中,在后续的研究中将考虑两者结合的措施,即同一组件同一时刻采取了 2 种或多种保障策略,在系统运行的过程中,考虑随时间变化的资源分配机制及相应的进化搜索算法.

参 考 文 献

[1] Zhou Jiang, Wang Weiping, Meng Dan, et al. Key technology in distributed file system towards big data analysis [J]. Journal of Computer Research and Development, 2014, 51(2): 382-394 (in Chinese)

(周江,王伟平,孟丹,等. 面向大数据分析的分布式文件系统关键技术[J]. 计算机研究与发展,2014,51(2):382-394)

[2] Amari S V, Dugan J B, Misra R B. Optimal reliability of systems subject to imperfect fault-coverage [J]. IEEE Trans on Reliability, 1999, 48(3): 275-284

[3] Du Xiaozhi, Qi Yong, Hou Di, et al. Software rejuvenation model based on reconfiguration and periodical rejuvenation [J]. Journal of Xi'an Jiaotong University, 2010, 44(1): 91-95 (in Chinese)
(杜小智,齐勇,侯迪,等. 重配置与周期再生相结合的软件再生模型[J]. 西安交通大学学报,2010,44(1):91-95)

[4] Nourelfath M, Chatelet E, Nahas N. Joint redundancy and imperfect preventive maintenance optimization for series-parallel multi-state degraded systems [J]. Reliability Engineering & System Safety, 2012, 103: 51-60

[5] Mi Haibo, Wang Huaimin, Yin Gang, et al. Resource on-demand reconfiguration method for virtualized data centers [J]. Journal of Software, 2011, 22(9): 2193-2205 (in Chinese)
(米海波,王怀民,尹刚,等. 一种面向虚拟化数字中心资源按需重配置方法[J]. 软件学报,2011,22(9):2193-2205)

[6] He Jialang, Zhang Kun, Meng Jin, et al. Hybrid software fault-tolerant model based on evolvable-modules redundancy [J]. Journal of Nanjing University of Science and Technology, 2012, 36(2): 272-277, 284 (in Chinese)
(何加浪,张琨,孟锦,等. 可进化模块冗余软件混合容错模型[J]. 南京理工大学学报,2012,36(2):272-277,284)

[7] Coit D W. Cold-standby redundancy optimization for nonrepairable systems [J]. IIE Transactions, 2001, 33(6): 471-478

[8] Jia Jia, Yang Xuejun, Li Zhiling. A redundancy-multithread-based multiple GPU copies fault-Tolerance technique [J]. Journal of Computer Research and Development, 2013, 50(7): 1551-1562 (in Chinese)
(贾佳,杨学军,李志凌. 一种基于冗余线程的GPU多副本容错技术[J]. 计算机研究与发展,2013,50(7):1551-1562)

[9] Zhu Jun, Guo Changguo, Wu Quanyuan. A runtime monitoring web services interaction behaviors method based on CPN [J]. Journal of Computer Research and Development, 2011, 48(12): 2277-2289 (in Chinese)
(朱俊,郭长国,吴泉源. 一种基于CPN的运行时监控服务交互行为的方法[J]. 计算机研究与发展,2011,48(12):2277-2289)

[10] Romera R, Valdes J E, Zequeira R I. Active-redundancy allocation in systems [J]. IEEE Trans on Reliability, 2004, 53(3): 313-318

[11] Ben Halima R, Fki E, Drira K, et al. A large-scale monitoring and measurement campaign for web services-based applications [J]. Concurrency Computation Practice and Experience, 2010, 22(10): 1207-1222

[12] He Pan, Yuan Yue, Wu Kaigui. Optimal multi-objective monitoring resources allocation in distributed systems [J]. Computer Science, 2014, 41(5): 64-67, 77 (in Chinese)
(何盼,袁月,吴开贵. 分布式系统监控资源多目标优化分配[J]. 计算机科学,2014,41(5):64-67,77)

[13] He Pan, Wu Kaigui, Wen Junhao, et al. Monitoring resources allocation for service composition under different monitoring mechanisms [C] //Proc of the 5th Int Conf on Complex, Intelligent and Software Intensive Systems. Los Alamitos, CA: IEEE Computer Society, 2011: 263-270

[14] He Pan. Research on resources allocation in distributed systems oriented at optimal reliability design [D]. Chongqing: Chongqing University, 2012 (in Chinese)
(何盼. 面向可靠性优化设计的分布式系统资源分配研究[D]. 重庆:重庆大学,2012)

[15] Chern M S. On the computational complexity of reliability redundancy allocation in a series system [J]. Operations Research Letters, 1992, 11(5): 309-315

[16] Coit D W. Maximization of system reliability with a choice of redundancy strategies [J]. IIE Transactions, 2003, 35(6): 535-543

[17] Tavakkoli-Moghaddam R, Safari J, Sassani F. Reliability optimization of series-parallel systems with a choice of redundancy strategies using a genetic algorithm [J]. Reliability Engineering and System Safety, 2008, 93(4): 550-556

[18] Chambari A, Rahmati S H A, Najafi A A, et al. A bi-objective model to optimize reliability and cost of system with a choice of redundancy strategies [J]. Computers and Industrial Engineering, 2012, 63(1): 109-119

[19] Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-Ⅱ [J]. IEEE Trans on Evolutionary Computation, 2002, 6(2): 182-97

[20] Chambari A, Najafi A A, Rahmati S H A, et al. An efficient simulated annealing algorithm for the redundancy allocation problem with a choice of redundancy strategies [J]. Reliability Engineering and System Safety, 2013, 119: 158-164

[21] Safari J. Multi-objective reliability optimization of series-parallel systems with a choice of redundancy strategies [J]. Reliability Engineering and System Safety, 2012, 108: 10-20

[22] Liu Y, Huang H Z, Wang Z, et al. A joint redundancy and imperfect maintenance strategy optimization for multi-state systems [J]. IEEE Trans on Reliability, 2013, 62(2): 368-378

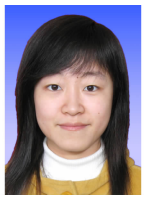
[23] Levitin G. Optimal structure of fault-tolerant software systems [J]. Reliability Engineering and System Safety, 2005, 89(3): 286-295

[24] Ahmadizar F, Soltanpanah H. Reliability optimization of a series system with multiple-choice and budget constraints using an efficient ant colony approach [J]. Expert Systems with Applications, 2011, 38(4): 3640-3646

[25] Zhuang Lu, Cai Mian, Shen Changxiang. Trusted dynamic measurement based on interactive Markov chains [J]. Journal of Computer Research and Development, 2011, 48(8): 1464-1472 (in Chinese)
(庄琭, 蔡勉, 沈昌祥. 基于交互式马尔可夫链的可信动态度量研究[J]. 计算机研究与发展, 2011, 48(8): 1464-1472)
[26] Krasnogor N, Smith J. A tutorial for competent memetic algorithms: Model, taxonomy, and design issues [J]. IEEE Trans on Evolutionary Computation, 2005, 9(5): 474-488
[27] Xia Y, Wang H P, Huang Y, et al. A stochastic model for workflow QoS evaluation [J]. Scientific Programming, 2006, 14(3/4): 251-265
[28] Wu Caihua, Liu Juntao, Peng Shirui, et al. Deriving Markov chain usage model from UML model [J]. Journal of Computer Research and Development, 2012, 49(8): 1811-1819 (in Chinese)
(吴彩华, 刘俊涛, 彭世蕤, 等. 基于 UML 的软件 Markov 链使用模型的构建[J]. 计算机研究与发展, 2012, 49(8): 1811-1819)
[29] Zitzler E, Deb K, Thiele L. Comparison of multiobjective evolutionary algorithms: Empirical results [J]. Evolutionary Computing, 2000, 8(2): 173-195
[30] Auger A, Bader J, Brockhoff D, et al. Theory of the hypervolume indicator: Optimal μ-distributions and the choice of the reference point [C] //Proc of the 10th ACM SIGRVO Conf on Foundations of Genetic Algorithms. New York: ACM, 2009: 87-102
[31] He P, Xie Q. Reliability analysis of service composition with service pools and optimal configuration of service pool size [G] //Principles, Methodologies, and Service-Oriented Approaches for Cloud Computing. Hershey, Pennsylvania: IGI Global, 2013: 344-370

He Pan, born in 1984. PhD and assistant researcher. Member of China Computer Federation. Her research interests include reliability optimization, reliable service computing and heuristic algorithm.

Tan Chun, born in 1974. Bachelor and engineer. His research interests include optimal reliability design, probability analysis, reliability and risk assessment (tanchun@cigit.ac.cn).



Yuan Yue, born in 1986. Master and assistant researcher. Her research interests include control theory and optimization problem.

Wu Kaigui, born in 1966. PhD, professor and PhD supervisor. His main research interests include fault-tolerant computing and distributed computing (kaiguiwu@cqu.edu.cn).