網格運算系統的資源配置最佳化設計

洪士程
朝陽科技大學資訊工程系
schong@cyut.edu.tw

林晉安
朝陽科技大學資訊工程系
s9827631@cyut.edu.tw

摘要

在本篇文章中，我們提出一個以序優化(OO)為基礎的演算法解決網格運算系統的資源配置最佳化問題，希望可以使服務可靠度最大化。首先提出一個近似模型在一個合理的時間內計算出資源配置的服務可靠度。接著利用所提出的演算法來解決資源配置最佳化問題。以序優化為基礎的演算法分為兩階段，在第一階段使用二進制粒子群最佳化(BPSO)演算法搭配近似模型來評估適應值並且選擇足夠好的解之子集合。然後在第二階段使用更精緻的近似模型進行目標搜索找出一個足夠好的解。我們以一個 8 節點為架構的網格運算系統作為模擬與測試範例，包含一個資源管理節點以及 11 條連結,所獲得足夠好的解在品質以及計算效能方面都有出色的表現。而利用 Pentium IV 電腦進行模擬，所提出的演算法僅花費 2.35 分鐘就可獲得足夠好的資源配置。

關鍵詞：二進制粒子群最佳化，序優化，網格運算系統，服務可靠度，資源配置。

Abstract

In this paper, we propose an ordinal optimization (OO) based algorithm for solving the resource allocation optimization problem of grid computing system to maximize the service reliability. An approximate model is firstly proposed to estimate the service reliability of a resource allocation design within a tolerable computation time. Next, we employ the proposed algorithm to solve the resource allocation optimization problem. The OO based algorithm consists of two stages. A binary particle swarm optimization (BPSO) algorithm is employed in the first stage using the approximate model for fitness evaluation and selects a subset of good enough solutions. Then, we proceed with the goal softening searching procedure in the second stage using more refined approximate models to search for a good enough solution. We have demonstrated the test results by simulating on an 8-node and 11-link grid computing system including one resource-managing node. The good enough solution obtained by the proposed algorithm is promising in the aspects of solution quality and computational efficiency. In addition, the proposed algorithm spends only 2.35 minutes in a Pentium IV PC to obtain the good enough resource allocation design.

Keywords: binary particle swarm optimization, ordinal optimization, grid computing system, service reliability, resource allocation.

1. 前言

網格運算系統(Grid computing system)已經被提出且應用在一個新的領域，有別於傳統的分散式運算系統，網格運算系統特別注重於大規模的資源分享、創新的應用[4,10]。當公司、顧客、政府和其他機構中在網格中經由連結，有效的利用資源時，網格就開始相當的令人關注，在網格原理的基礎下，而網格運算技術已經被提出且應用在現實生活中的特定問題，例如：解決複雜問題、啟動行動裝置、緊密協同合作等等。有別於分散式系統的高執行方針，網格運算系統屬於一種廣域分散式系統，例如在現實生活中調整資源分配與動態且大規模的分散式程式問題的處理。

網格服務可靠度定義為資源管理節點收到使用者的所要求的服務，蒐集遠端資源來協助完成，能成功執行程式在網格環境中的機率[2]。Dai 等人[5]提出了階層式模型在考慮不同類型的故障情形下分析及評價網格服務可靠度，如阻塞故障、逾時故障、匹配故障、網路故障、程式錯誤、資源故障等。Gholami 等人[6] 基於蒙地卡羅模擬方法提出一個演算法來估計網格運算系統的服務可靠度。然而，上述的方法在估計大型網格運算系統的服務可靠度時很耗費時間。為了克服這個缺點，本篇論文提出了一個近似模型在有限的計算時間內來估計服務可靠度。

資源配置最佳化問題屬於一個 NP-hard 的問題。在這類型的問題上典型的解決方法是使用基因演算法（GA）[12]，模擬退火法（SA）[14]，禁忌搜索法（TS）[7]，進化演算法（EA）

[1]和粒子群最佳化（PSO）演算法[11]。對於限制條件存在的問題利用TS和SA方法在處理上會有困難。與EA相比較，PSO能夠快速的收斂並從所有的粒子中找到最佳的解。相較於GA，PSO 演算法的優點在於容易實現並且需調整的參數比較少。然而，在 PSO 演算法更新程序期間確定粒子的可行性是不可或缺的。此外，在大型網格運算系統中計算粒子的適應值是非常耗時的，導致PSO演算法收斂非常緩慢。在一個大型的網格運算系統，由於計算每個資源配置的服務可靠度已是一個耗時的任務，欲使服務可靠度能夠達到最大，這可能會導致在有限的計算時間內無法獲得足夠好的解。縱使採用上述的啟發式搜尋方法在有限的計算時間內仍無法獲得一個足夠好的解，為了解決這個缺點，我們提出了一個序優化(OO)[8]為基礎的演算法來解決計算複雜度問題，序優化為基礎的演算法其概念是使用一個計算簡單得估計模型尋找出一個足夠好的解之子集合，而不是尋找全域最佳解。

本篇論文共分為五節，在第一節，簡單描述網格運算系統，在第二節中說明網格運算系統資源配置最化問題，希望使服務可靠度達到最大。第三節，提出一個近似模型來估計資源配置的服務可靠度。此外，我們利用序優化為基礎的演算法來解決資源配置最佳化問題。第四節，使用一個8節點及11連結的網格運算系統做為模擬範例及展示測試結果。最後，在第五節做出結論。

2. 問題與公式

假設網格運算系統包括 N 個節點以 N_i 表示，與 m 項計算資源以 R_j 表示。計算資源在有限的預算花費內，配置到可存取的節點而其計算資源的服務可靠度表示為 $R_s(\mathrm{x})$。在網格運算系統中，為了在有限的預算花費內購置必需的計算資源並且配置到適當的節點，可將資源配置最佳化問題表示為下列公式[3]。

$$\max \quad R_s(\mathbf{x}) \tag{1a}$$

$$\text{subject to} \quad \sum_{i=1}^{N} x_{i,j} \geq 1, \quad j = 1,...,m, \tag{1b}$$

$$x_{i,j} \leq 1, \quad i = 1,...,N, \quad j = 1,...,m, \tag{1c}$$

$$\sum_{j=1}^{m} x_{i,j} \leq \beta_i, \quad i = 1,...,N, \tag{1d}$$

$$\sum_{j=1}^{m} c_j \times \left(\sum_{i=1}^{N} x_{i,j} \right) \leq B. \tag{1e}$$

$R_s(\mathbf{x})$ 代表資源配置 $\mathbf{x}$ 的服務可靠度，$\mathbf{x} = [x_{i,j}]$，$i = 1,...,N$，$j = 1,...,m$，為一固定矩陣代表資源配置。$x_{i,j} = 1$ 表示計算資源 R_j 配置到節點 N_i，否則 $x_{i,j} = 0$；c_j 表示資源 R_j 的花費成本；B 代表購買計算資源的預算限制。不等式(1b)代表所有 m 項計算資源必須存在於網格運算系統中。不等式(1c)表示同一項計算資源不應該在同一節點配置超過一次。不等式(1d)表示在節點 N_i 的資源配置總數不應超過 β_i 的預定上限。不等式(1e)表示，配置的所有計算資源花費成本總和不應超出預算限制 B。

問題(1)是一個組合最佳化，以及NP-Hard問題。問題(1)的解搜尋空間的大小是 $2^{N \times m}$，這是一個巨大的空間當 N 和 m 的值是很大時。換句話說，問題(1)是要在搜尋空間大小為 $2^{N \times m}$ 且滿足所有限制的情況下找出一個解且能夠獲得最大的服務可靠度。然而，光檢查是否滿足所有限制就已不是一項容易的任務，此外，利用貝氏定理[2]在大型網格運算系統中計算資源配置的服務可靠度是非常耗時的。因此，為了解決問題(1)的計算複雜度，我們將採用序優化為基礎的演算法來解決所考慮的問題。

3. 序優化演算法

為了解決計算複雜度問題，我們提出了一個序優化為基礎的演算法來解決資源配置最佳化問題。提出的序優化為基礎的演算法分為兩個階段。第一階段是搜索階段，採用二進制粒子群最佳化(BPSO)演算法來搜索問題(1)的巨大解空間，接下來使用近似模型評估適應值然後找出 L_1 個足夠好的解。第二階段是開發階段，目標是從第一階段獲得的 L_1 個解，利用目標軟化搜索方法，在此階段使用更精細的近似模型找到一個足夠好的解。序優化主要的概念是使用簡單的估計模型來評估適應值並且選擇足夠好的解之子集合，然後再以更精細的近似模型，得到一個足夠好的解。接下來，首先介紹近似模型，然後以 BPSO 演算法處理可行性問題，最後呈現序優化為基礎的演算法。

3.1 使用近似模型評估服務可靠度

比照傳統的模型[2]，我們假設發生在節點和連結的故障是滿足波以松過程，且在不同元件發生的故障彼此是獨立的，這是一個合理的

假設因為在大規模網格運算系統中絕大多數的共享資源彼此之間是距離遙遠的。假定程式 P_m 可以在任何資源管理位置上執行。資源管理位置代表一個根節點 N_v，$\mathrm{M}_k(\mathrm{x})$ 代表第 k 條最小資源生成樹(MRST)。MRST 定義為一個資源生成樹 RST_k，沒有其他的資源生成樹 RST_j 是 RST_k 的子集合。x 表示資源配置，$\mathrm{P}[\mathrm{M}_k(\mathrm{x})]$ 表示服務可靠度。服務可靠度 $\mathrm{P}[\mathrm{M}_k(\mathrm{x})]$ 由以下公式表示。

$$\mathrm{P}[\mathrm{M}_k(\mathbf{x})] = e^{-\lambda_v \cdot (t(m)+T(v))} \times \prod_{L(i,j)\in \mathrm{M}_k(\mathbf{x})} e^{-\lambda_{i,j}\cdot T_c(i,j)} \times \prod_{j\neq v,\, j\in \mathrm{M}_k(\mathbf{x})} e^{-\lambda_j \cdot T(j)} \quad (2)$$

$L(i,j)$ 代表節點 N_i 與節點 N_j 之間的連結；$\lambda_{i,j}$ 代表 $L(i,j)$ 連結故障率；λ_j 代表 N_j 節點故障率；$t(m)$ 代表 P_m 所需的處理時間；$T_c(i,j)$ 代表通過連結 $L(i,j)$ 的總通信時間，可由以下公式計算。

$$T_c(i,j) = \frac{D(i,j)}{S(i,j)} \quad (3)$$

其中 $D(i,j)$ 為總資訊量是為了執行方案 P_m 在根節點 N_v 與資源 R_j 之間交換信號量 D_j，$S(i,j)$ 代表 $L(i,j)$ 連結的平均速度；$T(j)$ 代表節點 N_j 總通訊時間，可由以下公式計算。

$$T(j) = \sum_{i\in Q_j} T_c(i,j) \quad (4)$$

Q_j 代表在 $\mathrm{M}_k(\mathrm{x})$ 上與節點 N_j 之間的通訊的節點集合，$t(m)+\mathrm{T}(v)$ 代表在 P_m 上的總處理時間和節點 N_v 的總通訊時間。假設有 K 條 MRSTs 並給定資源配置 x，則該服務可靠度可由 $\mathrm{P}[\mathrm{M}_1(\mathrm{x})\cup \mathrm{M}_2(\mathrm{x})\cup \ldots \cup \mathrm{M}_k(\mathrm{x})]$ 計算出。

如上面所述，在大型網格運算系統要計算服務可靠度是非常耗時的。因此，我們建立一個近似模型來計算服務可靠度，而近似模型將被用在序優化為基礎的演算法上。這裡的近似模型並不使用全部 MRSTs，而是以有系統的方式採用較少條的 MRSTs，底下做一個簡略的說明。

演算法(一) 近似服務可靠度。

Step 1. 給定資源配置 x 共 K 條 MRSTs 的資料，$\mathrm{M}_1(\mathbf{x}),\ldots,\mathrm{M}_K(\mathbf{x})$。

Step 2. 計算 $\mathrm{P}[\mathrm{M}_1(\mathbf{x})],\ldots,\mathrm{P}[\mathrm{M}_K(\mathbf{x})]$，不失一般性情況下，假定 $\mathrm{P}[\mathrm{M}_1(\mathbf{x})]\geq \mathrm{P}[\mathrm{M}_2(\mathbf{x})]\geq \ldots \geq \mathrm{P}[\mathrm{M}_K(\mathbf{x})]$。

Step 3. 把 $\mathrm{M}_1(\mathbf{x})$ 當作第一條 MRSTs。

Step 4. 計算 $\mathrm{P}[\mathrm{M}_1(\mathbf{x})\cup \mathrm{M}_2(\mathbf{x})]$，$\mathrm{P}[\mathrm{M}_1(\mathbf{x})\cup \mathrm{M}_3(\mathbf{x})]$，..., $\mathrm{P}[\mathrm{M}_1(\mathbf{x})\cup \mathrm{M}_K(\mathbf{x})]$，不失一般性情況下，假定 $\mathrm{P}[\mathrm{M}_1(\mathbf{x})\cup \mathrm{M}_2(\mathbf{x})]$ 為最大值。接著把 $\mathrm{M}_2(\mathbf{x})$ 當作第二條 MRSTs。機率 $\mathrm{P}[\mathrm{M}_i(\mathbf{x})\cup \mathrm{M}_j(\mathbf{x})]$ 可以由下列公式計算出。

$$\mathrm{P}[\mathrm{M}_i(\mathbf{x})\cup \mathrm{M}_j(\mathbf{x})] = \mathrm{P}[\mathrm{M}_i(\mathbf{x})] + \mathrm{P}[\mathrm{M}_j(\mathbf{x})] - \mathrm{P}[\mathrm{M}_i(\mathbf{x})\cap \mathrm{M}_j(\mathbf{x})] \quad (5)$$

而機率 $\mathrm{P}[\mathrm{M}_i(\mathbf{x})\cap \mathrm{M}_j(\mathbf{x})]$ 可以由下列公式計算出。

$$\mathrm{P}[\mathrm{M}_i(\mathbf{x})\cap \mathrm{M}_j(\mathbf{x})] = e^{-\lambda_v \cdot (t(m)+T(v))} \times \prod_{L(i,j)\in \{\mathrm{M}_i(\mathbf{x})\cap \mathrm{M}_j(\mathbf{x})\}} e^{-\lambda_{i,j}\cdot T_c(i,j)} \times \prod_{j\neq v,\, j\in \{\mathrm{M}_i(\mathbf{x})\cap \mathrm{M}_j(\mathbf{x})\}} e^{-\lambda_j \cdot T(j)} \quad (6)$$

Step 5. 繼續進行上面的程序直到下式滿足

$$\mathrm{P}[\mathrm{M}_1(\mathbf{x})\cup \mathrm{M}_2(\mathbf{x})\cup \cdots \cup \mathrm{M}_i(\mathbf{x})] - \mathrm{P}[\mathrm{M}_1(\mathbf{x})\cup \mathrm{M}_2(\mathbf{x})\cup \cdots \cup \mathrm{M}_{i-1}(\mathbf{x})] \leq \varepsilon \quad (7)$$

其中 ε 為預先設定好的一個小的實數，代表近似的程度。$\mathrm{P}[\mathrm{M}_1(\mathbf{x})\cup \mathrm{M}_2(\mathbf{x})\cup \cdots \cup \mathrm{M}_i(\mathbf{x})]$ 將用來作為 $R_s(\mathbf{x})$ 的近似服務可靠度。

3.2 二進制粒子群最佳化演算法

在 BPSO 演算法中，每個粒子的位置代表二進制值 0 或 1[9]。BPSO 演算法初始化的粒子總數大小為 Ψ，每個初始粒子元素是隨機挑選的 0 和 1。每個粒子的值是可以改變的，一個粒子的速度被定義為一個機率可能會改變其粒子狀態。每個粒子在解空間內都會紀錄其座標資訊，直到找到本身最好的值(pbest)為止，同時透過粒子群最佳化找到全部最好的值(gbest) 。BPSO 演算法可以改變每個粒子的速度並向 pbest 和 gbest 位置靠近移動。

第 k 個粒子在解空間的位置矩陣和速度矩陣表示為 $\mathbf{X}^k=[x_{i,j}^k]$ 和 $\mathbf{V}^k=[v_{i,j}^k]$，$i=1,...,N$，$j=1,...,m$，$k=1,...,\Psi$，第 k 個粒子有自己的最佳位置矩陣，標示為 $\mathbf{Y}^k=[y_{i,j}^k]$，整體最好的位置矩陣標記為 $\mathbf{Y}^g=[y_{i,j}^g]$，表示在整個粒子群中發現最佳的粒子位置。

演算法(二) BPSO 演算法。

Step 1. 初始化。

(a) 設定常數 $t_{\max}$，w，c_1，c_2，$V_{\max}$。

(b) 初始化總數為 Ψ 的粒子位置 $\mathbf{Y}^g(0)=\mathbf{X}^1(0)$ 以及速度 $\mathbf{V}^k(0)$，其中 $x_{i,j}^k(0)\in[0,1]$ 而 $v_{i,j}^k(0)$ 全部設定為零，$i=1,...,N$，$j=1,...,m$，$k=1,...,\Psi$。

(c) 用近似模型評估每個粒子的近似值 $R_s(\mathbf{X}^k(0))$。設定 $t=0$，$\mathbf{Y}^k(0)=\mathbf{X}^k(0)$，$f_{best}^k=R_s(\mathbf{Y}^k(0))$，$\mathbf{Y}^g(0)=\mathbf{X}^1(0)$，$f_{best}^g=R_s(\mathbf{Y}^g(0))$.

Step 2. 更新速度以及位置。

(a) 每個粒子更新速度由下列公式表示:

$$v_{i,j}^k(t+1)=w\times v_{i,j}^k(t)+c_1\times r_1\times(y_{i,j}^k(t)-x_{i,j}^k(t))+c_2\times r_2\times(y_{i,j}^g(t)-x_{i,j}^k(t))$$
$$,i=1,...,N,\quad j=1,...,m,\quad k=1,...,\Psi. \qquad (8)$$

c_1 和 c_2 分別為認知參數和社會參數；w 代表慣性因子；r_1 和 r_2 代表獨立隨機變數均勻分佈在[0,1]範圍內，若 $v_{i,j}^k(t+1)>V_{\max}$，設定 $v_{i,j}^k(t+1)=V_{\max}$；若 $v_{i,j}^k(t+1)<-V_{\max}$，設定 $v_{i,j}^k(t+1)=-V_{\max}$。$V_{\max}$ 是允許的最大速度，控制解空間的搜索以及開發。

(b) 利用 sigmoid 函數計算門檻值:

$$s_{i,j}^k(t+1)=\frac{1}{1+e^{-v_{i,j}^k(t+1)}},\quad i=1,...,N,\quad j=1,...,m,$$
$$k=1,...,\Psi. \qquad (9)$$

(c) 粒子根據以下公式移動到新的位置:

$$x_{i,j}^k(t+1)=\begin{cases}1, & if \quad r_3<s_{i,j}^k(t+1)\\ 0, & otherwise\end{cases},$$
$$i=1,...,N,\ j=1,...,m,\quad k=1,...,\Psi. \qquad (10)$$

r_3 為一個隨機變數均勻分布在 0 到 1 範圍內。

Step 3. 評估。

對每個粒子使用近似模型評估適應函數 $R_s(\mathbf{X}^k(t+1))$。

Step 4. 修正 pbest 和 gbest。

(a) 若 $R_s(\mathbf{X}^k(t+1))\ge f_{best}^k$，則 $f_{best}^k=R_s(\mathbf{X}^k(t+1))$，$\mathbf{Y}^k(t+1)=\mathbf{X}^k(t+1),k=1,...,\Psi$.

(b) 若 $R_s(\mathbf{X}^k(t+1))\ge f_{best}^g$，則 $f_{best}^g=R_s(\mathbf{X}^k(t+1))$，$\mathbf{Y}^g(t+1)=\mathbf{X}^k(t+1)$，$k=1,...,\Psi$.

Step 5. 終止。

如果滿足停止條件，則終止，否則設定 $t=t+1$ 回到 Step 2 繼續執行。

對每個維度以及粒子將重複執行上述步驟，直到找到最好的解。如果目前的 $\mathbf{X}^k(t)$ 結果比 $\mathbf{Y}^k(t)$ 情況佳，將被存儲為最佳的個體狀態。對於上面所介紹所有方程式，有一些參數必須作出限制。隨機重量 c_1 和 c_2 被設定為 2.0，慣性因子 w 是設定為 1.0。最高限額 $V_{\max}$ 必須設定正確，以避免門檻過於接近 0.0 或 1.0。$V_{\max}$ 值通常設為 4.0[13]，使得具有最小機率為 $s(V_{\max})=0.018$ 可以改變位元的狀態。BPSO 演變過程是疊代的，直到足夠的疊代次數或是最小誤差已達到所需的性能指標。在我們的方法裡，BPSO 演算法將持續疊代，直到目前為止找到最佳的適應函數的最小增量，低於預定的正實數 ε_r 為止。

3.3 探索階段

在序優化為基礎的演算法的探索階段是最複雜的，因為在資源配置最佳化問題中初始解空間的候選解集合是巨大的。透過一個近似模型來評估服務可靠度，雖解決了計算最困難的部分，得以使全域搜尋技術看起來很容易。不過，對於 SA 和 TS 方法在處理有限制條件上是有困難的，以及要從初始的巨大解空間裡，必須找到一個足夠好的解之子集合。因此，BPSO 演算法可以處理有限制條件，在這個階段搭配近似模型應該是最好的搜尋方法。BPSO 演算法有兩個功能。首先，以近似模型評估適應函數可用來解決費時的評估問題，第二的功能是解的表示方法和修正程序，可用於處理解的可行性問題。在 BPSO 演算法，每一個粒子的位置矩陣表示為 $\mathbf{X}^k(t)$，對應到網格運算系統中 $\mathbf{x}$ 是資源配置。粒子更新後的新位置，可能導致 $\mathbf{x}$ 可能無法滿足問題（1）所有的限制。為了使 $\mathbf{x}$ 成為可行，將套用下列修正程序。

演算法(三) 修正程序

Step 1. 從 $j=1$ 開始檢查是否 $\sum_{i=1}^{N}x_{i,j}<1$，如果是，隨機選擇一個為零的 $x_{i,j}$ 並設定為 1，對 $j=2,\ldots,m$ 重複執行直到滿足不等式限制 (1b) 為止。

Step 2. 從 $i=1$ 開始檢查是否 $\sum_{j=1}^{m}x_{i,j}\le\beta_i$，如果不是，隨機選擇一個非零的 $x_{i,j}$ 並設定為 0，對 $i=2,\ldots,N$ 重複執行直到滿足不等式限制 (1d) 為止。

Step 3. 檢查是否 $\sum_{j=1}^{m}c_j\times\left(\sum_{i=1}^{N}x_{i,j}\right)\le B$，如果不

是，隨機選擇一個非零的$x_{i,j}$並設定為 0，對$i=2,\ldots,N$重複執行直到滿足不等式限制 (1e)為止。

3.4 開發階段

在序優化為基礎的演算法的開發階段，採用更精確的近似模型評估服務可靠度並選擇足夠好的解，以形成一個候選解之子集合。評估服務可靠度的近似值，是根據 3.1 節公式。因此， 在各階段逐漸減少近似程度值ε可使近似模型更完善，讓候選解之子集合逐漸縮小，最後階段即可選擇出足夠好的解。

在階段l的近似程度值定義為ε_l，$\varepsilon_l=\frac{1}{2}\varepsilon_{l-1}$(和$\varepsilon_l=\frac{1}{2^l}\varepsilon_0$)，$l=1,2,\ldots$。設定$\varepsilon_a=0.0001$，代表最後階段使用的模型是一個精確的模型。設定$L_1=\Psi$，在階段$l$裡足夠好的解之子集合大小設為$L_l=\lfloor L_{l-1}/2\rfloor$ (or $L_l=\lfloor\Psi/2^{l-1}\rfloor$)，$l=2,3,\ldots$。$\lfloor\bullet\rfloor$表示，如果得到的$L_l$值不是一個整數，四捨五入到最接近的整數。所有階段的總數定義為ℓ，可由以下公式獲得。

$$\ell=\arg\{\min_{\ell}(\varepsilon_\ell\le\varepsilon_a, L_\ell\le 1)\} \qquad (11)$$

上述公式表示ℓ值可以由下二個狀況挑選出最小值，(i)在子階段ℓ的近似值非常小時，即$\varepsilon_\ell\le\varepsilon_a$。(ii)最後挑選出來的候選解之子集合在最後子階段是最終解，即$L_\ell\le 1$。一旦ℓ確定，從子階段$l=1$到$\ell-1$，在階段l使用近似程度值$\varepsilon_l=\frac{1}{2^l}\varepsilon_0$，評估$L_l=\lfloor\Psi/2^{l-1}\rfloor$個候選解的目標函數值。依據目標函數值排列在階段$l$中的$L_l$個候選解，並選擇最好的$\lfloor\Psi/2^l\rfloor$候選解到下一個子階段$l+1$。在子階段$\ell$中獲得的唯一解就是最終的解。因此，序優化為基礎的演算法求解資源配置最佳化問題，做一個詳細說明如下。

演算法(四) 序優化為基礎的演算法

Step 1. 使用演算法(二)搭配大小為Ψ和演算法(一)搭配近似程度值ε_0解決資源配置最佳化問題，然後，從Ψ個候選解找出最好的L_1個解。

Step 2. 由公式(11)確定子階段ℓ數量。

Step 3. 從子階段$l=1$到$\ell-1$,使用更精確近似模型與$\varepsilon_l=\frac{1}{2^l}\varepsilon_0$評估$L_l$候選解的目標函數值;從這些$L_l$個候選解中選擇最佳$L_{l+1}$個解到子階段$l+1$。在$\ell$次子階段找出的最後解就是足夠好的解。

4. 測試結果

測試範例採用一個 8 個節點以及 11 個連結構成的網格運算系統，包括一個資源管理節點，如圖 1 所示，其中 N1 是資源管理的節點可以處理要求的程式。假設有一個程式P_m在網格運算系統中需要整合 4 種類型的計算資源（R1，R2，R3，R4）。因此，網格運算系統的搜尋解空間大小為$2^{8\times4}=4.3\times10^{10}$，雖然是很龐大的但很適合利用序優化為基礎的演算法尋求一個足夠好的資源配置。N1 的處理服務時間為 50 秒，預算限制 B 的總費用考慮二個預算，120K 以及 160K。表一為每個節點的故障率，表二為每個連結故障率以及頻寬。表三為各資源的資訊交換量以及花費成本。假設對於每個節點N_i的$\beta_i=2$。探索階段設定參數如下: 總數大小為$\Psi=50$，在 BPSO 演算法中近似模型評估每個適應值停止條件為$\varepsilon_r=0.001$和$\varepsilon_0=0.01$。在開發階段設置參數如下$L_1=\Psi$，$\varepsilon_a=0.0001$以及子階段數目為$\ell=7$。表四為在預算限制 B=120K 的限制下最有效的資源配置$\mathbf{x}^*$，最大化的服務可靠度為$R_s(\mathbf{x}^*)=0.9507$，購置資源花費為 118K。表五為在預算限制 B=160K 的限制下最有效的資源配置$\mathbf{x}^*$資源配置$\mathbf{x}^*$，最大化的服務可靠度為$R_s(\mathbf{x}^*)=0.9510$，購置資源花費為 158K。在使用 Pentium IV 電腦使用 Borland C++作模擬測試下，所耗費的 CPU 時間為 2.35 分鐘，是非常有效率的，使得提出的演算法可做即時性的運用。

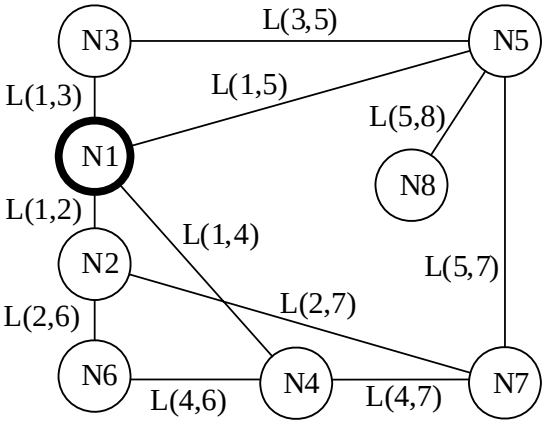

圖 1. 8 節點與 11 連結網格運算系統架構

表 1. 每個節點之故障率 λ_i (S^{-1})

節點	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8
λ_i	0.001	0.002	0.003	0.004	0.005	0.003	0.002	0.004

表 2. 每個連結的故障率 $\lambda_{i,j}$ (S^{-1})以及頻寬 $BW_{i,j}$ (KBPS)

連結	$\lambda_{i,j}$	$BW_{i,j}$
L(1,2)	0.001	20
L(1,3)	0.002	15
L(1,4)	0.002	10
L(1,5)	0.001	25
L(2,6)	0.005	30
L(2,7)	0.004	35
L(3,5)	0.006	40
L(4,6)	0.006	45
L(4,7)	0.005	50
L(5,7)	0.004	35
L(5,8)	0.005	45

表 3. 每個資源資訊交換量以及花費成本

資源	R_1	R_2	R_3	R_4
資訊交換量 D_j (Kbit)	300	200	100	50
花費成本 C_j ($)	12K	7K	14K	26K

表 4. 花費 118K 時的最佳資源配置

節點	N_1	N_2	N_3	N_4	N_5
資源配置	R_1, R_2	R_1, R_2	R_4	R_3, R_4	R_3

表 5. 花費 158K 時的最佳資源配置

節點	N_1	N_2	N_3	N_4	N_5	N_6
資源配置	R_1, R_2	R_3, R_4	R_2, R_4	R_4	R_1, R_3	R_3

5. 結論

在本篇論文中，提出以序優化為基礎的演算法解決資源配置最佳化問題，希望使網格運算系統的服務可靠度達到最大。首先提出一個近似模型在合理計算時間內來評估資源配置的服務可靠度，接著以序優化為基礎的演算法解決資源配置最佳化問題。測試範例使用 8 節點和 11 連結的網格運算系統，以 Pentium IV 電腦做模擬，所提出的演算法僅花費 2.35 分鐘就獲得足夠好的資源配置，而所獲得足夠好的解在品質以及計算效能方面都有出色的表現。

致謝

本論文係由國科會計劃 NSC99-2628-E-324-001 贊助。

參考文獻

[1] T. B¨ack and H. P. Schwefel, “An overview of evolutionary algorithms for parameter optimization,” ***Evol. Comput.***, vol. 1, no. 1, pp. 1-23, 1993.

[2] Y. S. Dai, M. Xie, and K. L. Poh, “Reliability analysis of grid computing systems,” **in *Proc. of the 2002 Pac. Rim Int. Symp. Depend. Comput.***, Tsukuba, Japan, 2002, pp. 97-104.

[3] Y. S. Dai and X. L. Wang, “Optimal resource allocation on grid systems for maximizing service reliability using a genetic algorithm,” ***Reliab. Eng. Syst. Saf.***, vol. 91, no. 9, pp. 1071-1082, Sep. 2006.

[4] Y. S. Dai, M. Xie, and K. L. Poh, “Availability modeling and cost optimization for the grid resource management system,” ***IEEE Trans. Syst. Man Cybern. Part A-Syst. Hum.***, vol. 38, no. 1, pp. 170-179, Jan. 2008.

[5] Y. S. Dai, Y. Pan, and X. Zou, “A hierarchical modeling and analysis for grid service reliability,” ***IEEE Trans. Comput.***, vol. 56, no. 5, pp. 681-691, May 2007.

[6] E. Gholami, A. M. Rahmani, and A. H. navin, “Using monte carlo simulation in grid computing systems for reliability estimation,” **in *Proc. of the IEEE Eighth Int. Conf. Netw.***, Gosier, France, 2009, pp. 380-384.

[7] A. R. Hedar and M. Fukushima, “Tabu Search directed by direct search methods for nonlinear global optimization,” ***Eur. J. Oper. Res.***, vol. 17, no. 3, pp. 329-349, Apr. 2006.

[8] Y. C. Ho, Q. C. Zhao, and Q. S. Jia, ***Ordinal optimization: Soft optimization for hard problems.*** Springer-Verlag, New York, 2007.

[9] J. Kennedy and R. C. Eberhart, “A discrete binary version of the particle swarm algorithm,” **in *Proc. of the 1997 IEEE Int. Conf. Syst. Man Cybern.: Comput. Cybern. Simul.***, Orlando, FL, USA, 1997, vol. 5, pp. 4104-4108.

[10] J. P. Sweeney and S. P. Ahuja, “Heuristic solutions to resource allocation in grid computing: a natural approach,” ***J. Supercomput.***, vol. 44, no. 2, pp. 179-198, May 2008.

[11] R. Poli, J. Kennedy, and T. Blackwell, “Particle swarm optimization: an overview,”

Swarm Intell., vol. 1, no. 1, pp. 33-57, Jun. 2007.

[12] S. N. Sivanandam and S. N. Deepa, ***Introduction to genetic algorithms.*** Heidelberg: Springer-Verlag, Berlin, 2008.

[13] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," **in** ***Proc. of the Seventh Ann. Conf. Evol. Prog. VII***, San Diego, CA, USA, 1998, pp. 591-600.

[14] B. W. Wah, Y. X. Chen, and T. Wang, "Simulated annealing with asymptotic convergence for nonlinear constrained optimization," ***J. Global Optim.***, vol. 39, no. 1, pp. 1-37, Sep. 2007.