人工知能学会研究会資料
SGI-FPAI-B503-05

クラウド上のソフトウェア要素最適配置問題の解法

Solving Optimal Software Component Configuration Problem in Cloud

田村直之 [1]　井上克巳 [2]　鍋島英知 [3]　番原睦則 [1]　宋剛秀 [1]

Naoyuki Tamura[1]　Katsumi Inoue[2]　Hidetomo Nabeshima[3]　Mutsunori Banbara[1]　Takehide Soh[1]

[1] 神戸大学 情報基盤センター

[1] Information Science and Technology Center, Kobe University

[2] 国立情報学研究所 情報学プリンシプル研究系

[2] Principles of Informatics Research Division, National Institute of Informatics

[3] 山梨大学大学院 医学工学総合研究部

[3] Department of Research Interdisciplinary Graduate School of Medicine and Engineering, University of Yamanashi

Abstract: Configuring complex networked applications by connecting software components distributed across multiple machines is a challenging problem, which requires a significant amount of expertise. We call the problem as Optimal software component configuration problem in this paper. A software named Zephyrus of Aeolus project is a tool of solving such problems. However, its performance is not satisfactory mainly because of its inappropriate constraint model. In this paper, we proposed a new constraint model which is much simpler than the previous model used in Zephyrus. We also confirmed the new model can be solved significantly faster than the previous model on a state-of-the-art constraint solver.

1　はじめに

アマゾン社の AWS (Amazon Web Services) などクラウド・サービスの利用が一般的になっている．大規模なソフトウェア・システムをクラウド上に構築する場合，必要なソフトウェア要素 (CMS サーバ，Web サーバ，DB サーバなど) をクラウド上の仮想マシン資源に適切に配置・相互接続し，さらに使用する資源を最小化する必要が生じる．本論文では，このような問題を**クラウド上のソフトウェア要素最適配置問題** (以下，単にソフトウェア要素最適配置問題) と呼ぶことにする．

パリ・ディドゥロ大学の Robert Di Cosmo 教授らは，この問題に関し Aeolus[*1] と呼ばれるオープンソースプロジェクトを開始している．Aeolus では，様々なプライベートクラウドおよびパブリッククラウドを対象とし，複数の仮想マシンを管理するソリューションの提供を目的としている．ソフトウェア要素最適配置問題を効率良く解くツールの開発は，Aeolus プロジェクトの主要な目標の一つである．

彼らは，論文 [1, 2] でこの問題を制約最適化問題として定式化し，**制約ソルバー**を用いて最適解を求める方法を提案した．しかし，数十個程度のソフトウェア要素から構成されるシステムの場合でも，彼らの開発したツール Zephyrus (制約ソルバーには Gecode[*2] を使用) では最適配置の求解に 30 分以上の時間を要しており，実用レベルに達しているとはいえない．

ソフトウェア要素最適配置問題の例として WordPress システムの配置問題を図 1 に示す．図中の (a) はソフトウェア要素の接続関係の条件を表しており，これらが入力として与えられる．図中の (b) は，与えられた条件を満たす配置のうちコストを最小にした配置結果である．結果では，4 つの仮想マシン上に HTTP load balancer, wordpress, mysql の 3 種類で合計 6 つのソフトウェア要素が配置されている．

この程度の規模ならば人手で最適配置 (あるいは最適に近い配置) を求めることは可能だが，より大規模になった場合，どのソフトウェア要素をどの仮想マシンに割当てるかだけでなく，ソフトウェア要素間の依存関係，インストールされている Linux ディストリビュー

[*1] http://www.aeolus-project.org

[*2] http://www.gecode.org

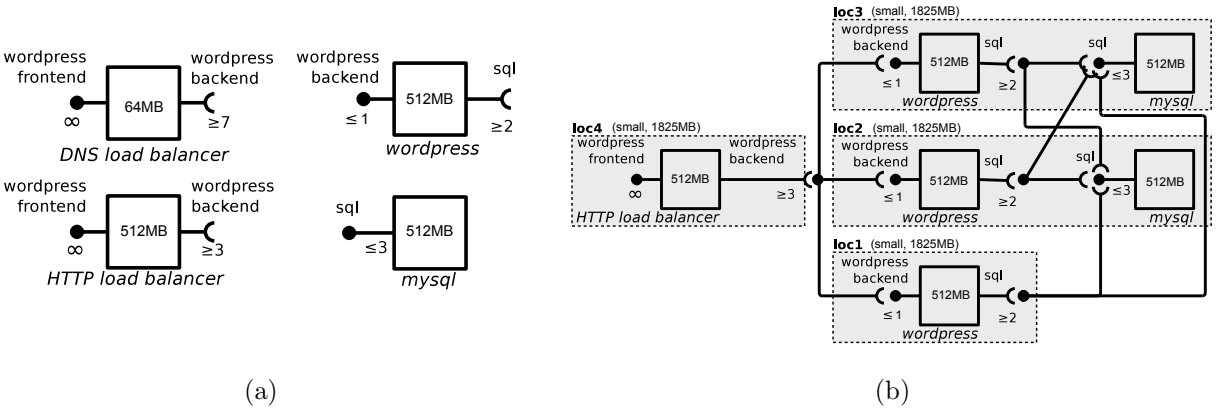

図 1　WordPress システムに対する最適配置の例

ションの種類なども考慮する必要があり，探索すべき空間は膨大になる．

したがってソフトウェア要素最適配置問題を制約最適化問題として定式化し (すなわち**制約モデル**を与え)，制約ソルバーを利用して解を求める方法は，有効な手段と考えられる．しかし前述のように，Zephyrus の性能は十分とはいえない．これは，用いている制約モデルが不適切な点が主な原因である．

そこで，本論文では Zephyrus の既存制約モデルを改善・改良した新しい制約モデルを提案する．また，制約ソルバー Diet-Sugar [*3] [8, 10, 9] を用い既存モデルと提案モデルの評価実験を行う．実験の結果，Zephyrus と比較し大幅な性能向上が可能であることを示す．

2　ソフトウェア要素最適配置問題

本節では，Zephyrus での定式化を元に，ソフトウェア要素最適配置問題について説明する．ただし簡単のためパッケージに関する部分は省略した．

マシンに配置される個々のソフトウェアをソフトウェア要素 (software component) あるいは単に**要素**と呼ぶ．要素はいくつかの種類 (WordPress など) に分類され，その種類をソフトウェア要素タイプ (software component type) あるいは単に**タイプ**と呼ぶ．タイプの集合を $\mathbf{T}$ で表す．タイプ毎に要素が必要とする資源量 (例えばメモリ使用量) が定まっている．また，いくつかのタイプについては配置する要素数の上限が定まっている．

要素間は同一名の**ポート** (port) を通じて接続される．ポート名の集合を $\mathbf{P}$ で表す．各要素は，0 個以上のポートを**要求** (require) および**供給** (provide) する．要求・供給するポートおよびその要求量・供給量はタイプ毎に定まっている．

[*3] http://kix.istc.kobe-u.ac.jp/~soh/dsugar/

要素を配置する個々のマシンを**場所** (location) と呼ぶ．場所の集合を $\mathbf{L}$ で表す．各場所には 1 つの**マシンタイプ**が割当てられる．マシンタイプの集合を $\mathbf{S}$ で表す．各マシンタイプに対し，利用可能な資源量 (例えばメモリ容量) およびコストが定まっている．ここでは，資源量としてメモリ容量のみを対象とするが，CPU 数などへの拡張も容易である．

図 1(a) の場合，タイプの集合 $\mathbf{T}$ は DNS load balancer, HTTP load balancer, wordpress, mysql の 4 つから成る．各々が必要とするメモリ使用量は箱の中に記入されている値であり，要求するポートと要求量は箱の右側に，供給するポートと供給量は箱の左側に記載されている．

形式的には，以下がソフトウェア要素最適配置問題に対する入力として与えられる．ただし $\mathbb{N} = \{0, 1, 2, \ldots\}$, $\mathbb{N}_\infty = \mathbb{N} \cup \{\infty\}$ である．

- $\mathbf{T}$: タイプの集合 (各要素はいずれかのタイプに分類される)
- $\mathbf{P}$: ポートの集合 (各要素はいくつかのポートを要求・供給する)
- $\mathbf{L}$: 場所の集合 (各要素はいずれかの場所に配置される)
- $\mathbf{S}$: マシンタイプの集合 (各場所にいずれかのマシンタイプが割当てられる)
- $\mathrm{max} \subset \mathbf{T} \times \mathbb{N}$: タイプ t に対して，要素を配置できる最大個数を $\mathrm{max}(t)$ として与える部分関数
- $\mathrm{Pro} \subset \mathbf{T} \times \mathbf{P} \times \mathbb{N}_\infty$: タイプ t の要素が供給するポートと供給量を $\mathrm{Pro}(t)$ として与える部分関数
- $\mathrm{Req} \subset \mathbf{T} \times \mathbf{P} \times \mathbb{N}$: タイプ t の要素が要求するポートと要求量を $\mathrm{Req}(t)$ としてを与える部分関数
- $\mathrm{mem} \subset \mathbf{T} \times \mathbb{N}$: タイプ t の要素が必要とするメモリ使用量を $\mathrm{mem}(t)$ として与える関数

- size $\subset \mathbf{S} \times \mathbb{N}$: マシンタイプ s のメモリ容量を size(s) として与える関数
- cost $\subset \mathbf{S} \times \mathbb{N}$: マシンタイプ s のコストを cost(s) として与える関数

ソフトウェア要素最適配置問題はこれらの入力に対し, 以下を満たす総コスト最小の解を求める問題である.

(1) 各場所に要素を配置する．同じ場所に配置した要素のメモリ使用量の合計は，その場所のマシンタイプのメモリ容量以下である．また同じタイプの要素の総数は，与えられた最大個数以下である．
(2) 要素間で同一名を持つ要求ポートと供給ポートを接続した時，各要素の各要求ポートに対する接続数が要求量以上，各要素の各供給ポートに対する接続数が供給量以下になっている．
(3) 要素が配置されている場所のマシンタイプのコストの総和を総コストとする．
(4) 別に与えられる仕様を満たす.

図 1 の例で, 別に与えられる仕様は以下のものである.

(1) 各場所に配置できる同じタイプの要素は 1 以下.
(2) ポート wordpress frontend が供給される.

この時の最適配置の例を図 1(b) に示す．6 つの要素が 4 つの場所に配置されており，要素のメモリ使用量の合計は各場所のメモリ容量以下になっている．また，総コストは最小である.

3 Zephyrus の制約モデル

Zephyrus [1, 2] の既存制約モデルを説明する．ただし簡単のためパッケージに関する制約は省略している.

3.1 要素に関する変数と制約

変数 N_{tsl} で，タイプ t の要素がマシンタイプ s の場所 l に配置される個数を表す．また，変数 N_t でタイプ t の要素が配置される総数を表す.

$$N_{tsl} \in \{0\,..\,\infty\} \qquad (t \in \mathbf{T},\ s \in \mathbf{S},\ l \in \mathbf{L}) \quad \text{(Z-T1)}$$
$$N_t \in \{0\,..\,\infty\} \qquad (t \in \mathbf{T}) \quad \text{(Z-T2)}$$
$$N_t \le \max(t) \qquad (t \in \mathbf{T},\ (t,\_) \in \max) \quad \text{(Z-T3)}$$
$$N_t = \sum_{s \in \mathbf{S},\ l \in \mathbf{L}} N_{tsl} \qquad (t \in \mathbf{T}) \quad \text{(Z-T4)}$$

3.2 ポートに関する変数と制約

変数 P_{psl} で，マシンタイプ s の場所 l でのポート p の供給量の合計を表す．変数 P_p でポート p の供給量の総量を表す.

$$P_{psl} \in \{0\,..\,\infty\} \qquad (p \in \mathbf{P},\ s \in \mathbf{S},\ l \in \mathbf{L}) \quad \text{(Z-P1)}$$
$$P_p \in \{0\,..\,\infty\} \qquad (p \in \mathbf{P}) \quad \text{(Z-P2)}$$
$$P_{psl} = \sum_{t \in \mathbf{T},\ (p,b) \in \mathrm{Pro}(t)} b\,N_t \qquad (p \in \mathbf{P},\ s \in \mathbf{S},\ l \in \mathbf{L}) \quad \text{(Z-P3)}$$
$$P_p = \sum_{s \in \mathbf{S},\ l \in \mathbf{L}} P_{psl} \qquad (p \in \mathbf{P}) \quad \text{(Z-P4)}$$

3.3 接続に関する変数と制約

変数 $B_{ptt'}$ は，ポート p でタイプ t の要素からタイプ t' の要素へ接続する個数を表す．この変数は，タイプ t でポート p が要求され，タイプ t' でポート p が供給される場合にのみ定義される.

$$B_{ptt'} \in \{0\,..\,\infty\} \qquad (p \in \mathbf{P},\ t,t' \in \mathbf{T},\ (t,p,\_) \in \mathrm{Req},\ (t',p,\_) \in \mathrm{Pro}) \quad \text{(Z-B1)}$$
$$a\,N_t \le \sum_{t' \in \mathbf{T},\ (p,\_) \in \mathrm{Pro}(t')} B_{ptt'} \qquad (p \in \mathbf{P},\ t \in \mathbf{T},\ (t,p,a) \in \mathrm{Req}) \quad \text{(Z-B2)}$$
$$b\,N_{t'} \ge \sum_{t \in \mathbf{T},\ (p,\_) \in \mathrm{Req}(t)} B_{ptt'} \qquad (p \in \mathbf{P},\ t' \in \mathbf{T},\ (t',p,b) \in \mathrm{Pro}) \quad \text{(Z-B3)}$$

制約 (Z-B2) は，タイプ t の要素の要求ポート p への総接続数が，それらの要求ポートの総要求量以上という条件を表している．制約 (Z-B3) も同様に，タイプ t の要素の供給ポート p からの総接続数が，それらの供給ポートからの総供給量以下という条件を表している.

3.4 資源に関する変数と制約

変数 S_{sl} はマシンタイプ s の場所 l が使用されているかどうかを表す．変数 M_{sl} はマシンタイプ s の場所 l のメモリ容量を表す.

$$S_{sl} \in \{0\,..\,1\} \qquad (s \in \mathbf{S},\ l \in \mathbf{L}) \quad \text{(Z-R1)}$$
$$S_{sl} = 1 \iff \sum_{t \in \mathbf{T}} N_{tsl} > 0 \qquad (s \in \mathbf{S},\ l \in \mathbf{L}) \quad \text{(Z-R2)}$$
$$M_{sl} = \mathrm{size}(s) \qquad (s \in \mathbf{S},\ l \in \mathbf{L}) \quad \text{(Z-R3)}$$
$$M_{sl} \ge \sum_{t \in \mathbf{T}} \mathrm{mem}(t)\,N_{tsl} \qquad (s \in \mathbf{S},\ l \in \mathbf{L}) \quad \text{(Z-R4)}$$

制約 (Z-R4) は，マシンタイプが s の場所 l のメモリ容量は，そこに配置されている要素の総メモリ使用量以上であるという条件を表している.

3.5 コストに関する変数と制約

変数 C は総コストを表す.

$$C \in \{0\,..\,\infty\} \quad \text{(Z-C1)}$$
$$C = \sum_{s \in \mathbf{S},\ l \in \mathbf{L}} \mathrm{cost}(s)\,S_{sl} \quad \text{(Z-C2)}$$
$$\text{minimize } C \quad \text{(Z-C3)}$$

3.6 WordPress 問題に関する制約

WordPress の例では，仕様としてさらに以下が指定される．ただし `wp_fe` はポート wordpress frontend を

表す．

$$N_{tsl} \le 1 \qquad (t \in \mathbf{T},\ s \in \mathbf{S},\ l \in \mathbf{L}) \qquad \text{(Z-W1)}$$
$$P_{\texttt{wp\_fe}} \ge 1 \qquad \text{(Z-W2)}$$

4 Zephyrus の制約モデルの改善

ここでは，元の Zephyrus の制約モデルを大きく変更せず，変数のドメインの縮小および GCD による係数の簡約化による改善方法について述べる．

4.1 変数のドメインの縮小

最大のメモリ容量を考慮すると，各場所におけるタイプ t の要素数を制限できる．その上限値を u_t とする．

$$N_{tsl} \le u_t \qquad (t \in \mathbf{T},\ s \in \mathbf{S},\ l \in \mathbf{L}) \qquad \text{(Z-T5)}$$
$$N_t \le u_t \cdot |\mathbf{L}| \qquad (t \in \mathbf{T}) \qquad \text{(Z-T6)}$$

ポートの供給量は無限大の場合がある．そのようなポートの集合を $\mathbf{Q}$ で表す．ポート $p \in \mathbf{Q}$ に対し，無限大であることを表す変数 Q_{psl}, Q_p を導入する．

$$Q_{psl} \in \{0\,..\,1\} \qquad (p \in \mathbf{Q},\ s \in \mathbf{S},\ l \in \mathbf{L}) \qquad \text{(Z-P5)}$$
$$Q_p \in \{0\,..\,1\} \qquad (p \in \mathbf{Q}) \qquad \text{(Z-P6)}$$

$$Q_{psl} = 1 \Longleftrightarrow \bigvee_{t \in \mathbf{T},\ (p,\infty) \in \mathrm{Pro}(t)} N_{tsl} > 0 \qquad (p \in \mathbf{Q},\ s \in \mathbf{S},\ l \in \mathbf{L}) \qquad \text{(Z-P7)}$$

$$Q_p = 1 \Longleftrightarrow \bigvee_{s \in \mathbf{S},\ l \in \mathbf{L}} Q_{psl} > 0 \qquad (p \in \mathbf{Q}) \qquad \text{(Z-P8)}$$

制約 (Z-P3) を以下で置き換える．

$$P_{psl} = \sum_{t \in \mathbf{T},\ (p,b) \in \mathrm{Pro}(t),\ b < \infty} b\, N_t \qquad (p \in \mathbf{P},\ s \in \mathbf{S},\ l \in \mathbf{L}) \qquad \text{(Z-P3}')$$

無限大の場合を特別視すれば，P_{psl} は有限の値とできるため，その上限値を u_p とする．

$$P_{psl} \le u_p \qquad (p \in \mathbf{P},\ s \in \mathbf{S},\ l \in \mathbf{L}) \qquad \text{(Z-P9)}$$
$$P_p \le u_p \cdot |\mathbf{L}| \qquad (p \in \mathbf{P}) \qquad \text{(Z-P10)}$$

また，接続に関する制約 (Z-B3) を以下で置き換える．

$$b\, N_{t'} \ge \sum_{t \in \mathbf{T},\ (p,\_) \in \mathrm{Req}(t)} B_{ptt'} \qquad (p \in \mathbf{P},\ t' \in \mathbf{T},\ (p,b) \in \mathrm{Pro}(t'),\ b < \infty) \qquad \text{(Z-B3}')$$

$$\bigvee_{t \in \mathbf{T},\ (p,\_) \in \mathrm{Req}(t)} B_{ptt'} > 0 \Longrightarrow N_{t'} > 0 \qquad (p \in \mathbf{P},\ t' \in \mathbf{T},\ (p,b) \in \mathrm{Pro}(t'),\ b = \infty) \qquad \text{(Z-B3}'')$$

最後に，WordPress 問題の仕様の制約 (Z-W2) を以下で置き換える．「有限の供給量が正になっているか，または無限の供給量がある」を意味する．

$$P_{\texttt{wp\_fe}} > 0 \ \vee\ Q_{\texttt{wp\_fe}} > 0 \qquad \text{(Z-W2}')$$

4.2 GCD による係数の簡約化

$\mathrm{mem}(t)$ の GCD を g_m とし，資源に関する制約 (Z-R3)(Z-R4) を以下で置き換える．

$$M_{sl} = \left\lfloor \frac{\mathrm{size}(s)}{g_\mathrm{m}} \right\rfloor \qquad (s \in \mathbf{S},\ l \in \mathbf{L}) \qquad \text{(Z-R3}')$$
$$M_{sl} \ge \sum_{t \in \mathbf{T}} \frac{\mathrm{mem}(t)\, N_{tsl}}{g_\mathrm{m}} \qquad (s \in \mathbf{S},\ l \in \mathbf{L}) \qquad \text{(Z-R4}')$$

cost の GCD を g_c とし，コストに関する制約 (Z-C2) を以下で置き換える．実際のコストの値は $g_\mathrm{c}\, C$ である．

$$C = \sum_{s \in \mathbf{S},\ l \in \mathbf{L}} \frac{\mathrm{cost}(s)\, S_{sl}}{g_\mathrm{c}} \qquad \text{(Z-C2}')$$

5 新しい制約モデルの提案

提案する新しい制約モデルについて説明する．

各場所は 1 つのマシンタイプしか取らない．そこで，場所 l の取るマシンタイプを表す変数 S_l を導入し，マシンタイプと場所の双方を添字としている変数からマシンタイプの添字を削除する．これにより，制約数が大幅に減少できる．

なお 4 章で述べた変数のドメインの縮小および GCD による係数の簡約化は，元の制約モデルと同様に適用可能である．

5.1 要素に関する変数と制約

変数 N_{tl} はタイプ t の要素が場所 l に配置される数を，変数 N_t はタイプ t の要素が配置される総数を表す．

$$N_{tl} \in \{0\,..\,\infty\} \qquad (t \in \mathbf{T},\ l \in \mathbf{L}) \qquad \text{(S-T1)}$$
$$N_t \in \{0\,..\,\infty\} \qquad (t \in \mathbf{T}) \qquad \text{(S-T2)}$$
$$N_t \le \max(t) \qquad (t \in \mathbf{T},\ (t,\_) \in \max) \qquad \text{(S-T3)}$$
$$N_t = \sum_{l \in \mathbf{L}} N_{tl} \qquad (t \in \mathbf{T}) \qquad \text{(S-T4)}$$

5.2 ポートに関する変数と制約

変数 P_{pl} は場所 l でのポート p の供給量を，変数 P_p はポート p の供給量の総量を表す．

$$P_{pl} \in \{0\,..\,\infty\} \qquad (p \in \mathbf{P},\ l \in \mathbf{L}) \qquad \text{(S-P1)}$$
$$P_p \in \{0\,..\,\infty\} \qquad (p \in \mathbf{P}) \qquad \text{(S-P2)}$$
$$P_{pl} = \sum_{t \in \mathbf{T},\ (p,b) \in \mathrm{Pro}(t)} b\, N_t \qquad (p \in \mathbf{P},\ l \in \mathbf{L}) \qquad \text{(S-P3)}$$
$$P_p = \sum_{l \in \mathbf{L}} P_{pl} \qquad (p \in \mathbf{P}) \qquad \text{(S-P4)}$$

5.3 接続に関する変数と制約

元の制約モデルと同一で良い．

5.4 資源に関する変数と制約

変数 S_l は場所 l で選択するマシンタイプを表す．ただし $S_l = 0$ で，場所 l が利用されないことを表す．変数 M_l は場所 l でのメモリ使用量を表す．

$$S_l \in \{0\} \cup \mathbf{S} \qquad (l \in \mathbf{L}) \quad \text{(S-R1)}$$

$$S_l > 0 \iff \sum_{t \in \mathbf{T}} N_{tl} > 0 \qquad (l \in \mathbf{L}) \quad \text{(S-R2)}$$

$$M_l \in \{0 \,..\, \max_{s \in \mathbf{S}}(\mathrm{size}(s))\} \qquad (l \in \mathbf{L}) \quad \text{(S-R3)}$$

$$M_l = \sum_{t \in \mathbf{T}} \mathrm{mem}(t)\, N_{tl} \qquad (l \in \mathbf{L}) \quad \text{(S-R4)}$$

$$S_l = 0 \Longrightarrow M_l = 0 \qquad (l \in \mathbf{L}) \quad \text{(S-R5)}$$

$$S_l = s \Longrightarrow M_l \leq \mathrm{size}(s) \qquad (l \in \mathbf{L},\ s \in \mathbf{S}) \quad \text{(S-R6)}$$

5.5 コストに関する変数と制約

グローバル基数制約 (global cardinality constraint; gcc) と呼ばれる制約を利用し，以下のように記述する．変数 X_s は，マシンタイプが s の場所の個数を表す．

$$C \in \{0 \,..\, \infty\} \qquad \text{(S-C1)}$$

$$X_s \in \{0 \,..\, |\mathbf{L}|\} \qquad (s \in \mathbf{S}) \qquad \text{(S-C2)}$$

$$\mathrm{gcc}(\{S_l \mid l \in \mathbf{L}\},\ \{(s, X_s) \mid s \in \mathbf{S}\}) \qquad \text{(S-C3)}$$

$$C = \sum_{s \in \mathbf{S}} \mathrm{cost}(s)\, X_s \qquad \text{(S-C4)}$$

$$\text{minimize } C \qquad \text{(S-C5)}$$

グローバル基数制約 $\mathrm{gcc}(\{x_1, \ldots, x_m\},\ \{(v_1, c_1), \ldots, (v_n, c_n)\})$ は，以下を意味する制約である．

$$c_j = \left|\{x \mid x \in \{x_1, \ldots, x_m\},\ x = v_j\}\right| \quad (1 \leq j \leq n)$$

すなわち，c_j は $x_i = v_j$ を満たす x_i の個数を表す．

5.6 WordPress 問題に関する制約

WordPress 問題の仕様は，以下のように表せる．

$$N_{tl} \leq 1 \qquad (t \in \mathbf{T},\ l \in \mathbf{L}) \qquad \text{(S-W1)}$$

$$P_{\texttt{wp\_fe}} \geq 1 \qquad \text{(S-W2)}$$

6 評価実験

評価実験により，以下の 4 つの制約モデルを比較する．

- **既存モデル**: 3 章で述べた Zephyrus の制約モデル
- **既存モデル改善版**: 既存モデルに 4 章の改善を行ったもの
- **提案モデル**: 5 章で述べた制約モデル
- **提案モデル改善版**: 提案モデルに 4 章の改善を行ったもの

実験には，図 2 に示す WordPress の問題を用いた．パラメータ W は wp のポート sql の要求量などを定めており，パラメータ L は場所の個数を定めている．W と L の値には表 1 に示す 5 通りを用いた．最大で 834 箇所の場所を必要とする大規模な問題となっている．

$$\begin{aligned}
\mathbf{T} &= \{\texttt{wp}, \texttt{mysql}, \texttt{http_lb}, \texttt{dns_lb}\} \\
\mathbf{P} &= \{\texttt{wp_fe}, \texttt{wp_be}, \texttt{sql}\} \\
\mathbf{L} &= \{0 \,..\, L-1\} \\
\mathbf{S} &= \{\texttt{small}, \texttt{medium}, \texttt{large}, \texttt{xlarge}\} \\
\max &= \{(\texttt{dns_lb}, 1)\} \\
\mathrm{Pro} &= \{(\texttt{wp}, \texttt{wp_be}, 1),\ (\texttt{mysql}, \texttt{sql}, 3), \\
&\quad\ (\texttt{http_lb}, \texttt{wp_fe}, \infty),\ (\texttt{dns_lb}, \texttt{wp_fe}, \infty)\} \\
\mathrm{Req} &= \{(\texttt{wp}, \texttt{sql}, W),\ (\texttt{http_lb}, \texttt{wp_be}, W), \\
&\quad\ (\texttt{dns_lb}, \texttt{wp_be}, 2W+1)\} \\
\mathrm{mem} &= \{(\texttt{wp}, 512),\ (\texttt{mysql}, 512), \\
&\quad\ (\texttt{http_lb}, 512),\ (\texttt{dns_lb}, 64)\} \\
\mathrm{size} &= \{(\texttt{small}, 1825),\ (\texttt{medium}, 4026), \\
&\quad\ (\texttt{large}, 8052),\ (\texttt{xlarge}, 16104)\}
\end{aligned}$$

図 2　実験に用いた WordPress 問題 (パラメータ W, L)

表 1　WordPress 問題のパラメータとその時の最小コスト

ポート要求量 W	場所数 L	最小コスト
10	34	2210
20	134	8710
30	300	19500
40	534	34710
50	834	54210

求解には，Diet-Sugar [8, 10, 9] および MiniSat+ [4] を用いた．Diet-Sugar は与えられた制約最適化問題を，順序符号化 [11] と対数符号化 [12] をハイブリッドした手法で命題論理の充足可能性問題 (SAT 問題) に変換し，SAT ソルバー [5, 7, 6] を用いて求解するシステムである．SAT ソルバーには MiniSat 2.2 [3] を用いた (MiniSat+ の内部で利用される)．符号化および求解を含めたタイムアウトを 1800 秒とし，比較対象の 4 モデルで最適解が求まるまでの CPU 時間を計測した．

最適解を求めるのに要した CPU 時間を表 2 に示す (単位は秒)．表中, TO はタイムアウト, MO はメモリーオーバーを意味する．実験した 5 種類の問題について，既存モデルは 1800 秒の時間制限以内に 1 問も最適解を求められなかったが，提案モデルは全問題について最適解を求めた．また 4 章で述べた 2 種類の改善を行うことにより，$W = 30, 40, 50$ という大規模な問題に対し 5 倍以上の速度向上が得られた．

表 2　WordPress 問題の最適解求解に要した CPU 時間 (単位: 秒) の比較

W	既存	既存改善版	**提案**	**提案改善版**
10	TO	196.98	7.68	2.53
20	TO	TO	34.89	11.97
30	TO	TO	117.57	23.65
40	MO	TO	425.62	80.63
50	TO	TO	1378.13	240.55

表 3　WordPress 問題を符号化した時に生成される節数 (単位: 千) の比較

W	既存	既存改善版	**提案**	**提案改善版**
10	1,604	306	238	25
20	24,736	2,953	1,623	226
30	–	12,667	5,838	955
40	–	–	15,433	2,876
50	–	–	33,255	6,740

表 3 に，Diet-Sugar および MiniSat+ が生成した SAT 問題の節数を示す (単位は千)．表中の “–” はタイムアウトまたはメモリーオーバーのため節数が不明なことを意味する．提案モデルおよび提案モデル改善版で，大幅に節数を削減できており，これが求解速度向上に寄与したと考えられる．

7　おわりに

本稿では，ソフトウェア要素最適配置問題について，既存の Zephyrus で用いられている制約モデルを改善・改良した新しい制約モデルを提案した．既存モデルと比較し，提案した制約モデルで大幅な性能向上が得られることを実験により確認した．

なお，本研究を進めるにあたっては平成 27 年度国立情報学研究所共同研究 (戦略研究公募型，研究題目「クラウド上のソフトウェア要素最適配置問題の解法」) の助成を受けた．

参考文献

[1] Roberto Di Cosmo, Michael Lienhardt, Jacopo Mauro, Stefano Zacchiroli, Gianluigi Zavattaro, and Jakub Zwolakowski. Automatic application deployment in the cloud: from practice to theory and back (invited paper). In *26th International Conference on Concurrency Theory (CONCUR 2015)*, pp. 1–16, 2015.

[2] Roberto Di Cosmo, Michael Lienhardt, Ralf Treinen, Stefano Zacchiroli, Jakub Zwolakowski, Antoine Eiche, and Alexis Agahi. Automated synthesis and deployment of cloud applications. In *ACM/IEEE International Conference on Automated Software Engineering (ASE '14)*, pp. 211–222, 2014.

[3] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT 2003), LNCS 2919*, pp. 502–518, 2003.

[4] Niklas Eén and Niklas Sörensson. Translating pseudo-Boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, Vol. 2, No. 1-4, pp. 1–26, 2006.

[5] 井上克巳, 田村直之. SAT ソルバーの基礎. 人工知能学会誌, Vol. 25, No. 1, pp. 57–67, 2010.

[6] 鍋島英知, 岩沼宏治, 井上克巳. GlueMiniSat 2.2.5: 単位伝搬を促す学習節の積極的獲得戦略に基づく高速 SAT ソルバー. コンピュータソフトウェア, Vol. 29, No. 4, pp. 146–160, 2012.

[7] 鍋島英知, 宋剛秀. 高速 SAT ソルバーの原理. 人工知能学会誌, Vol. 25, No. 1, pp. 68–76, 2010.

[8] Takehide Soh, Mutsunori Banbara, and Naoyuki Tamura. A hybrid encoding of CSP to SAT integrating order and log encodings. In *27th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2015)*, pp. 421–428, 2015.

[9] 宋剛秀, 番原睦則, 田村直之. 順序符号化と対数符号化を融合した制約充足問題のハイブリッド符号化. 日本ソフトウェア科学会第 32 回大会講演論文集, PPL6-3, 9 月 2015.

[10] 宋剛秀, 佐古田淳史, 番原睦則, 田村直之. 制約充足問題のハイブリッド符号化に向けて. 人工知能基本問題研究会, Vol. 97, pp. 65–73, 3 月 2015.

[11] Naoyuki Tamura, Akiko Taga, Satoshi Kitagawa, and Mutsunori Banbara. Compiling finite linear CSP into SAT. *Constraints*, Vol. 14, No. 2, pp. 254–272, 2009.

[12] 田村直之, 丹生智也, 番原睦則. 制約最適化問題と SAT 符号化. 人工知能学会誌, Vol. 25, No. 1, pp. 77–85, 2010.