数理解析研究所講究録
1489巻 2006年 233-238

近傍探索の解合流性に基づく並列局所探索法の考察

半田 祐一 (Yuichi Handa)* 小野 廣隆 (Hirotaka Ono)†
定兼 邦彦 (Kunihiko Sadakane)† 山下 雅史 (Masafumi Yamashita)†
*九州大学大学院システム情報科学府 †九州大学大学院システム情報科学研究院
*† Department of Computer Science and Communication Engineering,
Graduate School of Information Science and Electrical Engineering, Kyushu University

1 はじめに

NP困難な [2] 組合せ最適化問題を厳密に解くことは一般に難しいことが知られているが，これに対して近似解法を用いて解の最適性の保証はなくとも現実的な時間で比較的良質な解を求めるという立場がある [6]. このような(発見的) 近似解法設計でしばしば用いられるのが局所探索法 [1,3,7] である. 実際, タブー探索法, 遺伝的局所探索法, 模擬アニーリング法などの高精度近似解法は局所探索に基づくものと見なすことができる. 局所探索法はある解を変形することで得られる解の集合 (近傍) 内を探索し, 改善解を発見するとその改善解の近傍から同様に探索を行い, 近傍内に改善解がなくなるまでこれらの操作を繰り返す手法である.

一方，近年の計算機の普及，低価格化などから複数の計算機を用いることにより，計算困難な問題を現実的に解く並列化アルゴリズムの設計も広く研究されている.

本発表では，局所探索法の処理の高速化を目的とする並列局所探索アルゴリズムの設計法を提案する. 局所探索の並列化として，近傍探索を複数の計算機で行うことで探索時間を短くする並列化手法が考えられるが，単純な設計では局所探索の1度の解の改善ごとに計算機間で通信，待機が必要となり，オーバーヘッドが生じるために高速化を図ることは困難である. そこで我々は，局所探索法の持つ解の類似性を利用し高速化を図る並列化手法 (近傍分割併合法 [8] と呼ぶ) を提案する. 本手法では解空間の構造が以下で述べるような合流性を持つ場合，通信によるオーバーヘッドや無駄な探索の大幅な削減が期待できる.

本研究では, 組合せ最適化問題の中から, 多くの (逐次) 近似解法が提案されていて, ベンチマーク問題などが充実しているなどの理由で, 特に巡回セールスマン問題 [5] の2, Or-OPT近傍, LinKernighan近傍に基づく局所探索法に対して計算実験を行い, 提案する並列化手法の評価を行った.

2 準備

2.1 組合せ最適化問題

組合せ最適化問題の一般定義について述べる. 最適化問題は一般に以下のように表される.

$$\begin{array}{ll} \text{最小化} & f(\mathbf{x}) \\ \text{制約条件} & \mathbf{x} \in F \end{array}$$

f を目的関数, F を実行可能領域と呼ぶ. F は制約条件を解の全体を表し個々の解 $\mathbf{x} \in F$ を実行可能解, $\mathbf{x} \notin F$ を実行不可能解と呼ぶ. 目的関数 $f(\mathbf{x})$ は実数値あるいは整数値をとる関数

f: $F \to R$ (あるいは Z)

である. ただし, R は実数の集合, Z は整数の集合を表す. 目的関数 $f(\mathbf{x})$ を最小にする実行可能解を最適解 $\mathbf{x}_{opt} \in F$ と呼び，そのような解を見つけることが最適化問題の目標である.

2.2 巡回セールスマン問題

n 個の都市の集合 $V = \{1, \ldots, n\}$ と, 都市 i と都市 j の間の距離 $d_{i,j}$ が与えられているとき, 全ての都市をちょうど一度ずつ訪れて最初の都市に戻る巡回路の中で最短のものを求める問題が巡回セールスマン問題 (Travelling Salesman Problem) である. 実行可能領域 F は全都市を渡る巡回路 $\mathbf{x}$ の集合であり，解 $\mathbf{x}$ を n 個の都市の訪問順序，すなわち V の要素の順列 $(x_1, \ldots, x_n)$ とすると, 巡回路 $\mathbf{x}$ の総距離を示す目的関数 $f(\mathbf{x})$ は次のように書ける.

$$f(\mathbf{x}) = \sum_{k=1}^{n} d_{x_k, x_{(k+1)}} \qquad \text{ただし，} d_{x_n, x_{(n+1)}} = d_{x_n, x_1}$$

2.3 局所探索法

ある実行可能解 $\mathbf{x} \in F$ に対し，$\mathbf{x}$ に少しの変形を加えることで得られる解の集合 $N(\mathbf{x}) \subset F$ を $f(\mathbf{x})$ の近傍と呼び，また解 $f(\mathbf{x})$ から近傍 $N(\mathbf{x})$ 内の解を一つ生成するために $f(\mathbf{x})$ に加える変形操作を近傍操作と呼ぶ．局所探索法は，適当な解 $\mathbf{x}$ を初期解とし，$\mathbf{x}$ の近傍内に $\mathbf{x}$ の改善解 $\mathbf{x}'$(すなわち $f(\mathbf{x}') < f(\mathbf{x})$) があれば，$\mathbf{x} := \mathbf{x}'$ とする操作を，近傍内に改善解が存在しなくなるまで反復する手法である．$N(\mathbf{x})$ 内に改善解が存在しない $\mathbf{x}$ を局所最適解と呼ぶ．近傍 $N(\mathbf{x})$ 内には，一般には改善解が複数個存在するので，どの解を次の解として採用するかについては，戦略が可能である．このルールを移動戦略 (move strategy) という．代表的なものとして，次の 2 つがある．

・即時移動戦略：$N(\mathbf{x})$ 内で最初に発見した改善解に移動する．

・最良移動戦略：$N(\mathbf{x})$ 内を全て調べて，最良の改善解に移動する．

3 独立の定義

組合せ (最適化) 問題の解を表現する $\mathbf{x}$ は，その" 組合せ" という問題の性質より，n 個の変数 $X = \{x_1, ..., x_n\}$ を用いて $\mathbf{x} = (x_1, x_2, ..., x_n)$ と表現出来る．各 x_i に割り当てる値は，問題により自然数 N，もしくは $\{0, 1\}$ と様々である．本章では，共通要素を持たない 2 つの X の部分集合 $X_A = \{x_{a_1}, ..., x_{a_{|A|}}\}, X_B = \{x_{b_1}, ..., x_{b_{|B|}}\}$ の目的関数 f における独立性を定義する．

定義 3.0 $A = \{a_1, ..., a_{|A|}\}$ で定義される X の部分集合 $X_A = \{x_{a_1}, ..., x_{a_{|A|}}\}$ に対して，以下のように $\mathbf{x_A}, \mathbf{x_{\overline{A}}}$ を定義する．

$$\mathbf{x_A} = (\alpha_1, \alpha_2, ..., \alpha_n), \quad \alpha_i = \begin{cases} x_i & (i \in A) \\ 0 & otherwise \end{cases}$$

$$\mathbf{x_{\overline{A}}} = (\alpha_1, \alpha_2, ..., \alpha_n), \quad \alpha_i = \begin{cases} x_i & (i \notin A) \\ 0 & otherwise \end{cases}$$

$X_A \cap X_B = \phi$ のとき，解の表現 $\mathbf{x}$ は定義 3.0 で定義した $\mathbf{x_A}, \mathbf{x_B}, \mathbf{x_{\overline{A \cup B}}}$ を用いて $\mathbf{x} = \mathbf{x_A} + \mathbf{x_B} + \mathbf{x_{\overline{A \cup B}}}$ と書くことが出来る．

定義 3.1 $\mathbf{x}$ を n 個の変数 $X = \{x_1, ..., x_n\}$ で表すとき，X の部分集合 $X_A, X_B (X_A \cap X_B = \phi)$ は目的関数 f に対して独立である．

⇔

以下の式を満たす $g, h_A, h_B, h_{A \cup B}$ が存在する．

$$f(\mathbf{x}) = g(h_A(\mathbf{x_{\overline{A}}}), h_B(\mathbf{x_{\overline{B}}}), h_{A \cup B}(\mathbf{x_{\overline{A \cup B}}})) \qquad (1)$$

定理 3.1 X の部分集合 $X_A, X_B (X_A \cap X_B = \phi)$ が f に対して独立で，関数 g が以下のような和の式で表せるとき，

$$\begin{aligned} f(\mathbf{x}) &= g(h_A(\mathbf{x_{\overline{A}}}), h_B(\mathbf{x_{\overline{B}}}), h_{A \cup B}(\mathbf{x_{\overline{A \cup B}}})) \\ &= h_A(\mathbf{x_{\overline{A}}}) + h_B(\mathbf{x_{\overline{B}}}) + h_{A \cup B}(\mathbf{x_{\overline{A \cup B}}}) \end{aligned} \qquad (2)$$

X_A の割当が $\mathbf{x}$ と異なる解 $\mathbf{x_A}' + \mathbf{x_B} + \mathbf{x_{\overline{A \cup B}}}$，$X_B$ の割当が $\mathbf{x}$ と異なる解 $\mathbf{x_A} + \mathbf{x_B}' + \mathbf{x_{\overline{A \cup B}}}$ に対して以下が成立する．

$\mathbf{x_A} + \mathbf{x_B} + \mathbf{x_{\overline{A \cup B}}}$，$\mathbf{x_A}' + \mathbf{x_B} + \mathbf{x_{\overline{A \cup B}}}$，$\mathbf{x_A} + \mathbf{x_B}' + \mathbf{x_{\overline{A \cup B}}}$，$\mathbf{x_A}' + \mathbf{x_B}' + \mathbf{x_{\overline{A \cup B}}}$ が実行可能解

⇒

$$\begin{aligned} &f(\mathbf{x_A} + \mathbf{x_B} + \mathbf{x_{\overline{A \cup B}}}) - f(\mathbf{x_A}' + \mathbf{x_B} + \mathbf{x_{\overline{A \cup B}}}) \\ &= f(\mathbf{x_A} + \mathbf{x_B}' + \mathbf{x_{\overline{A \cup B}}}) - f(\mathbf{x_A}' + \mathbf{x_B}' + \mathbf{x_{\overline{A \cup B}}}) \end{aligned} \qquad (3)$$

かつ

$$\begin{aligned} &f(\mathbf{x_A} + \mathbf{x_B} + \mathbf{x_{\overline{A \cup B}}}) - f(\mathbf{x_A} + \mathbf{x_B}' + \mathbf{x_{\overline{A \cup B}}}) \\ &= f(\mathbf{x_A}' + \mathbf{x_B} + \mathbf{x_{\overline{A \cup B}}}) - f(\mathbf{x_A}' + \mathbf{x_B}' + \mathbf{x_{\overline{A \cup B}}}) \end{aligned} \qquad (4)$$

定理 3.1: 証明

式 (1), (2) より，以下が導かれる．

$$f(\mathbf{x_A} + \mathbf{x_B} + \mathbf{x_{\overline{A \cup B}}}) = h_A(\mathbf{x_{\overline{A}}}) + h_B(\mathbf{x_{\overline{B}}}) + h_{A \cup B}(\mathbf{x_{\overline{A \cup B}}}) \quad (5)$$

$$f(\mathbf{x_A}' + \mathbf{x_B} + \mathbf{x_{\overline{A \cup B}}}) = h_A(\mathbf{x_{\overline{A}}}) + h_B(\mathbf{x_{\overline{B}}}') + h_{A \cup B}(\mathbf{x_{\overline{A \cup B}}}) \quad (6)$$

$$f(\mathbf{x_A} + \mathbf{x_B}' + \mathbf{x_{\overline{A \cup B}}}) = h_A(\mathbf{x_{\overline{A}}}') + h_B(\mathbf{x_{\overline{B}}}) + h_{A \cup B}(\mathbf{x_{\overline{A \cup B}}}) \quad (7)$$

$$f(\mathbf{x_A}' + \mathbf{x_B}' + \mathbf{x_{\overline{A \cup B}}}) = h_A(\mathbf{x_{\overline{A}}}') + h_B(\mathbf{x_{\overline{B}}}') + h_{A \cup B}(\mathbf{x_{\overline{A \cup B}}}) \quad (8)$$

(5) − (6)

$$\begin{aligned} &f(\mathbf{x_A} + \mathbf{x_B} + \mathbf{x_{\overline{A \cup B}}}) - f(\mathbf{x_A}' + \mathbf{x_B} + \mathbf{x_{\overline{A \cup B}}}) \\ &= h_B(\mathbf{x_{\overline{B}}}') - h_B(\mathbf{x_{\overline{B}}}) \end{aligned} \qquad (9)$$

(7) − (8)

$$\begin{aligned} &f(\mathbf{x_A} + \mathbf{x_B}' + \mathbf{x_{\overline{A \cup B}}}) - f(\mathbf{x_A}' + \mathbf{x_B}' + \mathbf{x_{\overline{A \cup B}}}) \\ &= h_B(\mathbf{x_{\overline{B}}}') - h_B(\mathbf{x_{\overline{B}}}) \end{aligned} \qquad (10)$$

式 (9), (10) より，式 (4) が導かれる．同様に，(5) − (7) と (6) − (8) より式 (5) が導かれる．(証明終)

この定理は，$\mathbf{x_A}$ の割当を変更する近傍操作 i_A は，$\mathbf{x}$ に対しても，また $\mathbf{x_B}$ の割当を変更する近傍操作 i_B を行った解 $i_B(\mathbf{x}) = \mathbf{x_A} + \mathbf{x_B}' + \mathbf{x_{\overline{A \cup B}}}$ に対しても，同程度の改善であることを示す．このとき，2つの近傍操作 i_A，i_B が項書換え系の合流性のような性質を持つと考えることが出来る．

多くの組合せ最適化問題に対して合流性をもつような解表現が可能である．具体例として，以下の小節で巡回セールスマン問題独立性を述べる．

3.1 巡回セールスマン問題の解の独立性

2.2 小節で用いたように，n 個の都市の集合 V の要素の順列を実行可能解を表現するとき，解を表現する $\mathbf{x}$ は n 個の変数 $X = \{x_1, ..., x_n\}$ を用いて $\mathbf{x} = (x_1, x_2, ..., x_n)$ と表現出来る．実行可能解は順回パスを表し，x_i と $x_{i+1}(1 \leq i \leq n, x_{n+1} = x_1)$ の示す都市の間にはパスが存在する．

以下の式を満たす X の部分集合 X_A, X_B を考える．

$$
\begin{aligned}
&X_A = \{x_{a_1}, ..., x_{a_{|A|}}\}, X_B = \{x_{b_1}, ..., x_{b_{|B|}}\} \subset X \\
&X_A \cap X_B = \phi \\
&1 \leq i \leq |A| - 1,\ a_{i+1} - a_i = 1 \text{ or } 1 - n \\
&1 \leq i \leq |B| - 1,\ b_{i+1} - b_i = 1 \text{ or } 1 - n \\
&b_1 - a_{|A|} \neq 1 \text{ or } 1 - n \\
&a_1 - b_{|B|} \neq 1 \text{ or } 1 - n
\end{aligned}
$$

これは順回路 $(x_1, x_2, ..., x_n)$ 中に，都市 $x_{a_1}, ..., x_{a_{|A|}}$ の順にと繋ぐパス $(x_{a_1}, ..., x_{a_{|A|}})$ と，都市 $x_{b_1}, ..., x_{b_{|B|}}$ の順に繋ぐパス $x_{b_1}, ..., x_{b_{|B|}}$ が存在し，またその2つのパスは互いに重複せず，2つのパスの間には少なくとも1つは都市が挟まれていることを意味する．

順回路 $(x_1, x_2, ..., x_n)$ からこの 2 つのパスを取り除くことで生じる 2 つのパスをそれぞれ $(x_{s_1}, ..., x_{s_{|S|}})$ と $(x_{t_1}, ..., x_{t_{|T|}})$ とする (図 1).

順回路 $\mathbf{x}$ はこの 4 つのパスを結合したものなので，目的関数 f を以下のように書き換えることが出来る．

$$
\begin{aligned}
f(\mathbf{x}) = d_{x_{t_{|T|}}, x_{a_1}} &+ \sum_{k=1}^{|A|-1} d_{x_{a_k}, x_{a_{k+1}}} \\
+ d_{x_{a_{|A|}}, x_{s_1}} &+ \sum_{k=1}^{|S|_1-1} d_{x_{s_k}, x_{s_{k+1}}} \\
+ d_{x_{s_{|S|}}, x_{b_1}} &+ \sum_{k=1}^{|B|-1} d_{x_{b_k}, x_{b_{k+1}}} \\
+ d_{x_{b_{|B|}}, x_{t_1}} &+ \sum_{k=1}^{|T|_1-1} d_{x_{t_k}, x_{t_{k+1}}}
\end{aligned}
\tag{11}
$$

式 (11) の第 1 項〜第 3 項の和は X_B に依存しないので $h_B(\mathbf{x_{\overline{B}}})$ に，第 5 項〜第 7 項の和は X_A に依存しないので $h_A(\mathbf{x_{\overline{A}}})$ に，第 4 項と第 8 項の和は X_A と X_B に依存しないので $h_{A \cup B}(\mathbf{x_{\overline{A \cup B}}})$ に置き換える．

$$
f(\mathbf{x}) = h_A(\mathbf{x_{\overline{A}}}) + h_B(\mathbf{x_{\overline{B}}}) + h_{A \cup B}(\mathbf{x_{\overline{A \cup B}}}) \tag{12}
$$

式 (12) は定義 3.1 の独立の定義に一致するので，X_A と X_B は独立である．

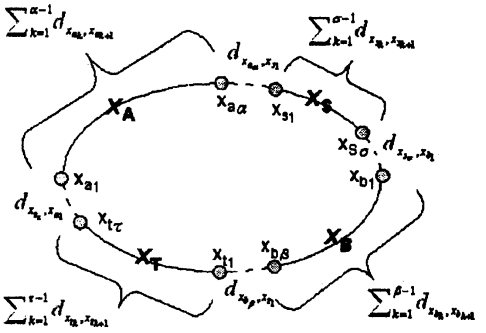

図 1: 実行可能解 $\mathbf{x}$ の構造

4 並列アルゴリズム: Neighbourhood Composition

1 台のマスタと N 台のスレーブを用いた並列アルゴリズムを示す．

アルゴリズム マスタ

level[i] : i 番目のスレーブ S_i の状態 $(0 \leq i \leq N)$

- level[i] = "former"　　S_i は以前の解を探索中
- level[i] = "current"　　S_i は現在の解を探索中
- level[i] = "wait"　　S_i は待機中

step1. 初期解 $\mathbf{x}$ を生成し、$\mathbf{x}$ の近傍 $N(\mathbf{x})$ を N 個に分割し $(N_1(\mathbf{x}),...,N_N(\mathbf{x}))$，$N$ 台のスレーブ S_i に $\mathbf{x}$ と $N_i(\mathbf{x})$ を送信.

step2. 受信待ち．あるスレーブ S_k からメッセージ M を受信したら，
・M が"No Sol" であれば，step3. へ進む.
・M が近傍操作 i であれば，step4. へ進む.

step3. 送ってきたスレーブ S_k の level が，
・level[k]="former"なら，level[k] を"current"にして、$\mathbf{x}$ と $N(\mathbf{x})$ を分割した $N_k(\mathbf{x})$ を送信し，step2. に戻る。
・level[k]="current"なら，level[k] を"wait"にする．この時点で全てのスレーブの level が"wait"であれば $\mathbf{x}$ を出力して終了．そうでないなら step2. に戻る.

step4. ・受信した近傍操作 i が現在の解 $\mathbf{x}$ を改善出来れば，つまり $f(i(x)) > f(x)$ であれば，step5. へ進む.
・受信した近傍操作 i で $\mathbf{x}$ を改善出来なければ step6. へ進む.

step5. 近傍操作 i で現在の解 $\mathbf{x}$ を $x := i(x)$ と更新する．level が"current"である全てのスレーブを level="former"に．level が"wait"である全てのスレーブを"current"にして，$\mathbf{x}$ と $N(\mathbf{x})$ を分割した $N_i(\mathbf{x})$ を送信し，step6. へ進む.

step6. S_k の level[k] を"current"にして，$\mathbf{x}$ と $N(\mathbf{x})$ を分割した $N_k(\mathbf{x})$ を送信し，step2. へ戻る.

アルゴリズム スレーブ S_i

step1. 受信待ち．マスタから $\mathbf{x}$ と $N_i(\mathbf{x})$ を受信したら step2. へ進む.

step2. $N_i(\mathbf{x})$ 内を改善解を求めて探索.
・改善解 x' が見つかれば，近傍操作 $i : x \to x'$ をマスタに送信し，step1. へ戻る.
・改善解が $N_i(\mathbf{x})$ 内に無ければ，メッセージ "No Sol" をマスタに送信，step1. へ戻る.

N 個の変数 *level* はこのアルゴリズムを止めるためのものであるので，*level* の操作以外のアルゴリズムを説明する.

具体的に巡回セールスマン問題 (TSP) を解く，2 辺を切って繋ぎかえるという 2-OPT 近傍の操作を用いた局所探索法の並列化を考えてみる.

図 2 の 1 段目のように，現在の解 $\mathbf{x_0}$ の近傍を 2 台のスレーブ S_0, S_1 が探索しているとする．S_0 が $\mathbf{x_0}$ のツアーを $c-h, d-g$ を切って，$c-d, g-h$ と繋ぎかえるという近傍操作 (= reverse$(x_2 - x_3)$) で改善解 $\mathbf{x_1}$ が得られることを発見し，改善操作 i_0 をマスタに送信し，マスタは i_0 に従って暫定解を $i_0(\mathbf{x_1}) = \mathbf{x_1}$ に更新する.

ここでマスタは暫定解 $\mathbf{x_1}$ を S_0 にのみ送信する．この時点で S_1 は過去の解 $\mathbf{x_0}$ を探索していることになるが (図 2 の 2 段目)，例えば，S_1 が $\mathbf{x_0}$ のツアーを $a-f, b-e$ を切って，$a-b, e-f$ と繋ぎかえるという近傍操作 (= reverse$(x_6 - x_7)$) で改善解 $\mathbf{x_1'}$ が得られることを発見したとき (図 2 の 3 段目)，S_1 はこの $\mathbf{x_0}$ から $\mathbf{x_1'}$ への近傍操作 i_1 をマスタに送信するが，i_1 は以下のように $\mathbf{x_1}$ を $\mathbf{x_2}$ へ改善することが可能である.

$$i_1(\mathbf{x_1}) = i(b,e,a,f,g,h,d,c) = (b,a,e,f,g,h,d,c)$$

図より $f(\mathbf{x_1}) > f(\mathbf{x_2})$ なのでこの近傍操作は改善である.
i_0 : reverse$(x_2 - x_3)$, i_1 : reverse$(x_6 - x_7)$
$X_A = \{x_2, x_3\}$, $X_B = \{x_6, x_7\}$

$$\begin{aligned} f(\mathbf{x}) &= \sum_{k=1}^{8} d_{x_k, x_{k+1}} \\ &= \sum_{k=1}^{3} d_{x_k, x_{k+1}} + d_{x_4, x_5} \\ &+ \sum_{k=5}^{7} d_{x_k, x_{k+1}} + d_{x_8, x_1} \end{aligned} \tag{13}$$

式 (13) の第 1 項の和は X_B に依存しないので $h_B(\mathbf{x_{\overline{B}}})$ に，第 3 項の和は X_A に依存しないので $h_A(\mathbf{x_{\overline{A}}})$ に，第 2 項と第 4 項の和は X_A と X_B に依存しないので $h_{A \cup B}(\mathbf{x_{\overline{A \cup B}}})$ に置き換える.

$$f(\mathbf{x}) = h_A(\mathbf{x_{\overline{A}}}) + h_B(\mathbf{x_{\overline{B}}}) + h_{A \cup B}(\mathbf{x_{\overline{A \cup B}}})$$

これは X_A と X_B が独立であることの必要十分条件である．よって 2 つの近傍操作 i_0, i_1 は解 $\mathbf{x_0}$ の独立な変数 X_A, X_B の操作であり，これにより過去の解への近傍操作 i_1 が現在の解 $i_0(\mathbf{x_0})$ にも適用できたわけである.

このように，全てのスレーブが現在の解を探索する必要はなく，よって本並列化手法，近傍分割併合法では探索作業を分割するだけでなく，改善解を発見したスレーブ以外への通信をカットすることで，通信によるオーバーヘッドを削減することを目指したものである.

近傍分割併合法で局所探索法の処理時間がどの程度短縮できるかは，解の独立性，つまり解く問題(の目的関数)と局所探索で用いる近傍構造(解をどのように変形するか)に大きく依存する．代表的な組合せ最適化問題の一つである巡回セールスマン問題に対して，様々な近傍構造に基づく局所探索法を近傍分割併合法により並列化した実験結果を次節で示す．

5 実験結果と考察

下記の性能の計算機を用いて実験を行った．

CPU: Pentium4 2.26GHz
MEM: 512MB

使用する問題例は，巡回セールスマン問題の問題例が公開されているウェブサイト TSPLIB [1]より入手した att532, pr1002, nrw1379 を利用する．下記のような近傍を探索する 3 つのプログラムの並列化を行った．

A. 2-OPT 近傍を探索
B. 2-OPT と Or-OPT を探索 [2]
C. 5-Lin-Kernighan 近傍を探索 [3]

A, B, C とも順回路のうち複数の辺を切断して，順回路になるように繋ぎかえる操作を近傍操作としており，A は 2 本の辺，B は最大 3 本の辺，C は最大 5 本の辺を一度の近傍操作で切断し，繋ぎかえる．

表 1 は元のプログラムと並列化したプログラムの実行時間を表している．どのプログラムに対してもスレーブを増やすごとに実行時間が短縮されているのがわかる．しかし，1) スレーブを多くすると台数効果が低下し，また 2)"A", "B" のプログラムに比べると，"C" の並列化の効果が小さい．1) の理由としてはスレーブが増加すると改善の適用の失敗する割合が大きくなることが考えられる．表 2 はスレーブから受信した改善の適用の失敗率を表している．スレーブの台数が多いと失敗率が大きいことが分かる．これはスレーブとマスタの持つ解構造の近似性ががスレーブを多くすると共に薄れるのが原因と考えられる．また表 2 より，"C" での失敗率は"B" より大きい．これは 1 つの近傍操作で切る辺の本数は"C" の近傍では最大 5 本対し，"B" では高々3 本であるので，1 つの近傍操作により割当を変化させる変数の数が"C" は "B" より多く，よって解の独立性が小さいのが原因で，これが 2) の結果を引き起こしていると考えられる．

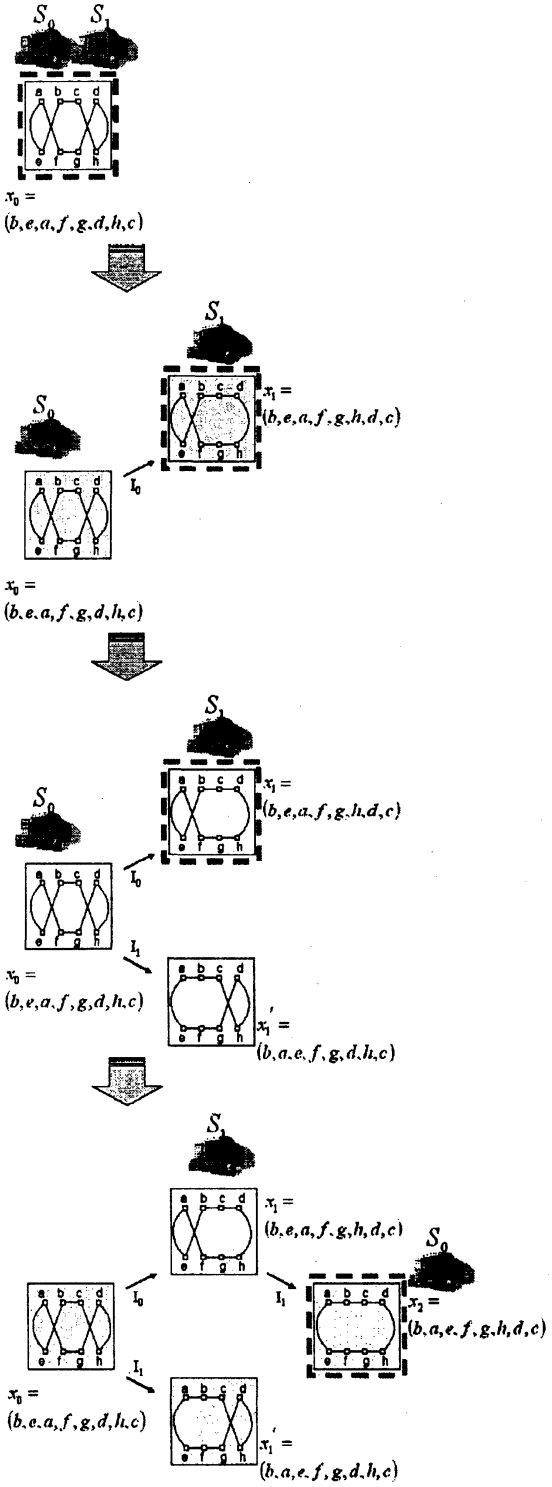

図 2: 例:TSP を解く 2OPT 近傍の局所探索法の並列化

[1] http://www.crpc.rice.edu/softlib/tsplib/
[2] http://ist.ksc.kwansei.ac.jp/%7Eibaraki/today/tsp1.c
[3] http://tcslab.csce.kyushu-u.ac.jp/%7Eu1/old_htmlprogram/lkh3_1.c

instance	近傍	original	2 slaves	4 slaves	8 slaves
att532 (532cities)	2OPT	5.26(1.00)	3.61(0.68)	2.73(0.51)	2.43(0.46)
	2,Or-OPT	62.83(1.00)	35.98(0.57)	25.16(0.40)	20.36(0.32)
	5-LK-OPT	4.70(1.00)	4.30(0.91)	2.85(0.60)	2.54(0.54)
pr1002 (1002cities)	2OPT	23.36(1.00)	14.23(0.60)	10.06(0.43)	8.25(0.35)
	2,Or-OPT	349.42(1.00)	199.03(0.56)	118.58(0.34)	83.19(0.24)
	5-LK-OPT	21.68(1.00)	17.79(0.82)	12.66(0.58)	9.57(0.44)
nrw1379 (1379cities)	2OPT	38.61(1.00)	25.49(0.66)	18.64(0.48)	16.11(0.41)
	2,Or-OPT	881.49(1.00)	494.68(0.56)	309.04(0.35)	203.28(0.23)
	5-LK-OPT	35.42(1.00)	30.23(0.85)	24.41(0.68)	17.78(0.50)

表 1: Average run time (s) and its ratio to the original of parallelized "2OPT", "2,Or-OPT" and "k-LK-OPT"

近傍	改善	2slaves	4slaves	8slaves
2,OrOPT	2edges	2.4	4.8	8.9
	3edges	13.5	33.2	42.9
	total	2.6	5.4	9.6
5LK-OPT	2edges	10.9	26.0	41.5
	3edges	12.7	28.3	44.6
	4edges	13.6	29.7	47.3
	5edges	16.9	37.3	55.4
	total	12.80	29.3	46.0

表 2: 改善の失敗率 (%)(Instance = att532)

6 まとめと今後の課題

局所探索法の解を表す複数の変数からなる集合 X の部分集合の目的関数における独立性を定義し，この独立性から並列度を抽出し，局所探索法の高速化を目的とする並列化手法である近傍分割併合法を提案した．また，代表的な組合せ最適化問題である巡回セールスマン問題を解く局所探索法に対して本手法で並列化する計算実験を行い，局所探索法の処理時間の短縮に成功した．

巡回セールスマン問題以外の組合せ最適化問題に対する計算実験や，局所探索法で用いる近傍構造の性質から，ある程度失敗率を予測し，本並列化手法の効果を見積もれるようにすることが今後の課題である．

参考文献

[1] E. Aarts and J. K. Lenstra, *Local Search in Combinatorial Optimization*, John Wiley & Son, 1997.

[2] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, 1979.

[3] P.E. Gill, W. Murray and M.H. Wright, *Practical Optimization*, Academic Press. 1981.

[4] K.Helsgaun, An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic, *European Journal of Operational Research* 126 (1), 106-130, 2000.

[5] E. L. Lawler, J. K. Lenstra, A. H. G Rinnooy Kan and D. B. Shmoys, The Traveling Salesman Problem, *A Guided Tour of Combinatorial Optimization*, John Wiley and Sons 1985.

[6] V. V. Vazirani, *Approximation Algorithms*, SpringerVerlag, 200.

[7] M. Yagiura and T. Ibaraki, On Metaheuristic Algorithms for Combinatorial Optimization Problems, *Systems and Computers in Japan*, 32 (3), 33-55, 2001.

[8] Y. Handa, H. Ono, K. Sadakane, M. Yamashita, Neighborhood Composition: A Parallelization of Local Search Algorithms, 11th European PVM/MPI Users' Group Meeting Proceedings (Lecture Notes in Computer Science), 155-163, September 2004.