2E5-OS-09b-2

SATによる車両運用計画問題の定式化と集中／分散解法の適用

Centralized and Distributed SAT for the Rolling Stock Operation Problem

下牧 昌太郎*1　尾松 郁哉　平山 勝敏
Shotaro SHIMOMAKI　Ikuya OMATSU　Katsutoshi HIRAYAMA

神戸大学大学院海事科学研究科
Graduate School of Maritime Sciences, Kobe University

The rolling stock operation involves the problem of assigning a fleet of trains to a given operation timetable. This problem has been typically formulated as the 0-1 integer programming problem to be solved by a MIP solver. In this paper, we first encode this problem as SAT and apply some state-of-the-art SAT solvers. We then explore the possibility to apply distributed SAT (DisSAT) solving when multiple train companies try to assign their trains to partially-shared operation timetables. We found that SAT/DisSAT approach is promising through our experimental evaluation using realistic instances.

1. はじめに

車両運用計画問題は，鉄道ダイヤ（運用）に対して車両（編成）を割り当てる問題である．近年，情報通信技術の発展に伴い，運用計画を自動作成するシステムの導入が進んでいるが，あくまで支援にとどまっており，最後は熟練技術者が作成しているのが現状である．

本稿では，車両運用計画問題を充足可能性判定問題（SAT）として定式化し，SAT ソルバーを用いて解く新たな解法を提案する．SAT は，CNF 命題論理式の充足可能性を判定する問題であり，近年，さまざまな応用問題を SAT に定式化し高速な SAT ソルバーで解くというアプローチが成功を収めており，特に注目を集めている [井上 10]．

さらに，複数の鉄道会社が運用を一部共有して相互直通運転を行なう際の車両運用計画問題（分散車両運用計画問題）を節集合分割型分散 SAT で定式化して既存アルゴリズムである M-ABTS [下牧 12] で解く．この解法では，鉄道会社間で共有される共同運行区間の運用情報のみを交換するだけでよく，各鉄道会社は既存のシステムを活用することができる．

2. 車両運用計画問題

車両運用計画問題は，鉄道ダイヤ（運用）に対して車両（編成）を割り当てる問題である．この問題は，ダイヤ担当者が定めた所与の運行計画のすべての運用に対して，種々の制約を考慮しながら，各運用に 1 つずつ編成を割り当てる制約充足問題と見なすことができる．

一方，複数の鉄道会社が運用を一部共有して相互直通運転を行なう場合には，各鉄道会社は自社の運用に編成を割り当てると同時に，共同運行区間の運用にどの会社の編成を割り当てるかを鉄道会社間で解決する必要がある．その際，平時には担当者による事前の話し合いで解決可能だが，急な車両故障が発生する等の緊急時には迅速に対応することが困難だと予想される．今後，利用者の利便性向上のために相互直通運転を実施する鉄道会社が増えると予想されるが，関係する鉄道会社の既存システムを通信回線で接続して，平時および緊急時の対応を分散システムにより解決することは非常に有用だと考えられる．本稿では，この問題設定を特に分散車両運用計画問題とよぶ．

なお，車両運用計画問題を扱った従来研究としては，機関車と客車の割り当てコストを数理計画モデルを用いて最小化するもの [Cordeau 00]，制約充足に基づく解法でグリーディ探索とバックトラック探索を組み合わせたもの [大槻 10]，まず一日分の車両運用計画を求め，次に検査制約を満たしながらそれをローテーションさせた複数日の計画を求めるもの [今泉 10] 等，国内外で多数の事例があるが，筆者らの知る限り分散車両運用計画問題を扱った研究は皆無である．

2.1 定義

車両運用計画問題において各鉄道会社は編成と運用の情報を保持する．編成には計画初日の最初に出庫すべき場所（初日出庫場所）ならびに計画最終日の最後に入庫すべき場所（最終日入庫場所）が定められている．一方，運用には出庫場所ならびに入庫場所と，出庫時刻ならびに入庫時刻が定められている．すなわち，場所の集合を R，時刻の集合を T，編成の集合を H，運用の集合を O として，

- $h_i \in H$：計画期間中に利用可能な編成番号 i の編成
- $s_i \in R$：編成 h_i の初日出庫場所
- $t_i \in R$：編成 h_i の最終日入庫場所
- $o_j \in O$：運用番号 j の運用．なお，その運用における出庫場所を $depP_j \in R$，入庫場所を $arvP_j \in R$，出庫時刻を $depT_j \in T$，入庫時刻を $arvT_j \in T$ とする

と表記する．

また，各編成 h_i に対して運用の系列（運用パス）を定義し，P_i と表記する．運用パス P_i は次の制約をすべて満たすものとする．

- つなぎ制約：運用パス P_i は，編成 h_i の初日出庫場所 s_i をスタートし，かつ，系列中で連続する 2 つの運用は時間と場所が接続していなければならない．
- 最終日制約：運用パス P_i は，計画最終日における運用の結果，最終日入庫場所 t_i に入庫しなければならない．

連絡先: 平山勝敏，神戸大学大学院海事科学研究科，〒 658-0022 神戸市東灘区深江南町 5-1-1，hirayama@maritime.kobe-u.ac.jp

*1 現在，株式会社日立情報制御ソリューションズ

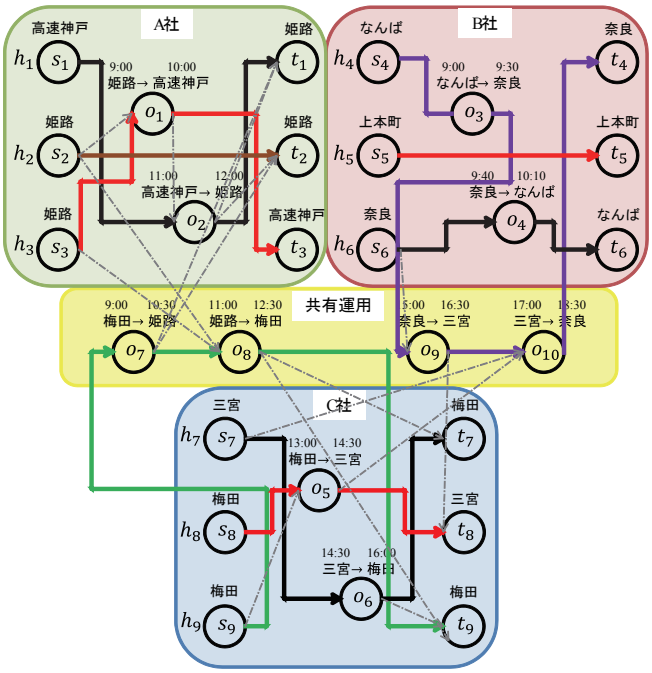

図 1: 3 社間の運用ネットワーク

- 運用限定制約：車両性能や検査日程等の都合により，運用パス P_i はある運用 o_j を含むことができない．

図 1 は相互直通運転を実施する 3 社間の分散車両運用計画問題の運用ネットワークの例である．運用ネットワークでは，運用がノードであり，任意の 2 つの運用 o_m と o_n がつなぎ制約を満たすとき（すなわち，$depP_m$ が $arvP_n$ 一致し，かつ，$depT_m$ が $arvT_n$ より早いとき），ノード o_m と o_n の間に有向アーク $(o_m, o_n) \in E$ が張られる．図中で色付きの実線で示された有向アークはある編成の運用パスを表している．一方，点線の有向アークは，どの編成の運用パスにも含まれていないことを示している．

また，運用には，各鉄道会社が自社の編成を割り当てる自社運用と，相互直通運転区間に対応しており自社あるいは他社の編成を割り当てる共有運用の 2 種類がある．図 1 では，運用 o_1, o_2 は A 社の自主運用，運用 o_3, o_4 は B 社の自主運用，運用 o_5, o_6 は C 社の自主運用であり，一方，運用 o_7, o_8 は A 社と C 社による共有運用，運用 o_9, o_{10} は B 社と C 社による共有運用である．

車両運用計画問題とは，編成の集合 H と運用の集合 O を所与として運用ネットワークを構成し，そのすべての運用ノードをちょうど 1 度ずつ訪問するような運用パス（解）を求める問題である．一方，分散車両運用計画問題は，上記に加えて鉄道会社の集合 A を所与として，図 1 のようないわゆる分散運用ネットワークを構成し，すべての自主運用と共有運用をちょうど 1 度ずつ訪問するような運用パス（解）を求める問題である．

2.2 節集合分割型分散 SAT

本研究では，分散車両運用計画問題を節集合分割型分散 SAT として定式化する．SAT は，与えられた CNF 命題論理式に対して，その値を真にするような変数への真偽値の割り当てが存在するかどうかを判定する問題である．真または偽の値をとる変数 x とその否定 $\neg x$ を総称してリテラルとよび，このリテラルを論理和で結合したものを節，節を論理積で結合したものを CNF 式とよぶ．すなわち CNF 式は節集合と見なせる．この節集合を複数のエージェントが分割して所有し，エージェント間で共有される変数には同じ値を割り当てるという制約を課した SAT を特に節集合分割型分散 SAT とよぶ．図 1 は，共有運用に関する変数を複数エージェントで共有し，各社の車両運用計画問題の制約をエージェント内の節集合で表現した分散 SAT として記述できる．

問題を節集合分割型分散 SAT として定式化した後は，既存アルゴリズムである M-ABTS [下牧 12] を用いて解く．M-ABTS は，エージェント間の非同期型メッセージ通信により節集合分割型分散 SAT を解くアルゴリズムであり，有限時間内で解を発見するか，あるいは，解がないことを発見して停止する．紙面の都合上，M-ABTS の詳細は [下牧 12] に譲り，本稿では定式化のみ説明する．

2.3 定式化

与えられた運用ネットワークに対して，各編成に対する運用パスが分散車両運用計画問題の解となるための制約条件を SAT の節で表現する．そのために本稿では以下の変数を導入する．

- x_j^i：運用 o_j で編成 h_i を利用するとき真（T），そうでないとき偽（F）となる．
- y_{mn}：運用 o_m と運用 o_n 間の有向アークが，いずれかの編成の運用パスに含まれるとき真（T），そうでないとき偽（F）となる．
- y_{sj}^i：編成 h_i の初日ノード s_i と運用 o_j 間の有向アークが編成 h_i の運用パスに含まれるとき真（T），そうでないとき偽（F）となる．
- y_{jt}^i：運用 o_j と編成 h_i の最終日ノード t_i 間の有向アークが編成 h_i の運用パスに含まれるとき真（T），そうでないとき偽（F）となる．
- y_i：編成 h_i の初日出庫場所 s_i と最終日入庫場所 t_i が同じ場合に，編成 h_i をどの運用にも利用しないとき真（T），そうでないとき偽（F）となる．
- z_j^k：共有運用 o_j で会社番号 k の編成を利用するとき真（T），そうでないとき偽（F）となる．

以下，図 1 の A 社の車両運用計画問題を用いて制約の具体例を示す．

2.3.1 初日ノードに関する制約

各編成 h_i の初日ノード s_i に関する制約は次の 1 つのみである．

制約 1 各編成 h_i の初日ノード s_i から出る h_i の運用パスはちょうど 1 本である．

例えば，図 1 の A 社の初日ノード s_2 からは，それぞれノード o_1, t_2, o_8 へ 3 本の有向アークが出ているが，h_2 の運用パスに含まれるのは 1 本だけなので，

$$AtLeastOne(y_{s1}^2, y_2, y_{s8}^2) \equiv y_{s1}^2 \vee y_2 \vee y_{s8}^2 \quad (1)$$

$$AtMostOne(y_{s1}^2, y_2, y_{s8}^2) \equiv \begin{cases} \neg y_{s1}^2 \vee \neg y_2 \\ \neg y_{s1}^2 \vee \neg y_{s8}^2 \\ \neg y_2 \vee \neg y_{s8}^2 \end{cases} \quad (2)$$

という制約を A 社は所有する．AtMostOne 制約は与えられた変数のうち高々1 つの変数が真となるという制約であり，逆に AtLeastOne 制約は与えられた変数のうち少なくとも 1 つの変数が真となるという制約である．編成 h_2 に対して，これら 2 つの制約を同時に満たすことを制約 1 は要請している．

なお，一般に AtLeastOne 制約は 1 つの節で記述できるが，AtMostOne 制約は，例えば $x_1, \cdots, x_n$ と n 個の変数が与えられると $n(n-1)/2$ 個の節で記述する必要があり，n が大きくなると節の数が莫大になる．そこで，本研究では [Sinz 05] で提案されている AtMostOne 制約に対する効率的な符号化法を採用している．この符号化法では，n が 6 以上のとき補助変数 $\alpha_1, \cdots, \alpha_{n-1}$ を導入し，AtMostOne 制約を

$$(\neg x_1 \vee \alpha_1) \wedge (\neg x_n \vee \neg \alpha_{n-1}) \wedge \bigwedge_{1<i<n} \Big((\neg x_i \vee \alpha_i) \wedge (\neg \alpha_{i-1} \vee \alpha_i) \wedge (\neg x_i \vee \neg \alpha_{i-1}) \Big) \tag{3}$$

とすることにより節の数を大幅に削減できる．

2.3.2 運用ノードに関する制約

運用ノード o_i に関する制約は次のとおりである．

制約 2 自社運用について，自社の編成を 1 つだけ割り当てる．一方，共有運用については，どちらか一方の会社の編成を 1 つだけ割り当てる．ただし，運用限定制約で禁止されている編成は割り当てない．

例えば，図 1 の共有運用 o_7 では，A 社（会社番号 1）あるいは C 社（会社番号 3）のどちらかが編成を割り当てるため，両者は

$$AtLeastOne(z_7^1, z_7^3) \equiv z_7^1 \vee z_7^3 \tag{4}$$

$$AtMostOne(z_7^1, z_7^3) \equiv \neg z_7^1 \vee \neg z_7^3 \tag{5}$$

という制約を所有する．また，A 社は自身が編成を割り当てる場合に備え，

$$\neg z_7^1 \vee AtLeastOne(x_7^1, x_7^2, x_7^3) \tag{6}$$

$$\neg z_7^1 \vee AtMostOne(x_7^1, x_7^2, x_7^3) \tag{7}$$

$$(\neg x_7^1 \vee z_7^1) \wedge (\neg x_7^2 \vee z_7^1) \wedge (\neg x_7^3 \vee z_7^1) \tag{8}$$

という制約をもつ（C 社も同様）．式（6）と（7）は A 社が運用 o_7 に編成を割り当てる場合，編成 h_1, h_2, h_3 のいずれか 1 つが割り当てられることを意味している．また，式 (8) は運用 o_7 に編成 h_1, h_2, h_3 のいずれかが割り当てられるならば，A 社が運用 o_7 を担当することを意味している．

制約 3 自社運用について，そこから出る運用パスはちょうど 1 本である．一方，共有運用については，自社の編成を割り当てるとき，そこから出る運用パスはちょうど 1 本である．

例えば，図 1 の共有運用 o_7 からは，最終日ノード t_1, t_2，および，運用ノード o_8 への有向リンクがあるため，A 社は

$$\neg z_7^1 \vee AtLeastOne(y_{7t}^1, y_{7t}^2, y_{78}) \tag{9}$$

$$\neg z_7^1 \vee AtMostOne(y_{7t}^1, y_{7t}^2, y_{78}) \tag{10}$$

という制約をもつ（C 社も同様）．

制約 4 各有向アークについて，それが運用パスに含まれるならば，その両端のノードには同じ編成を割り当てる．

例えば，図 1 の有向アーク (o_7, o_8) に関して，A 社は

$$\neg x_7^1 \vee \neg y_{78} \vee x_8^1 \tag{11}$$

$$\neg x_7^2 \vee \neg y_{78} \vee x_8^2 \tag{12}$$

$$\neg x_7^3 \vee \neg y_{78} \vee x_8^3 \tag{13}$$

という制約をもつ（C 社も同様）．なお，式（11）は，運用 o_7 に編成 h_1 が割り当てられ，かつ，有向アーク (o_7, o_8) が運用パスに含まれるならば，運用 o_8 には編成 h_1 を割り当てることを意味している．

制約 5 自社運用について，そこに入る運用パスはちょうど 1 本である．一方，共有運用については，自社の編成を割り当てるとき，そこに入る運用パスはちょうど 1 本である．

例えば，図 1 の共有運用 o_8 へは，初日ノード s_2, s_3，および，運用ノード o_7 からの有向リンクがあるため，A 社は

$$\neg z_8^1 \vee AtLeastOne(y_{s8}^2, y_{s8}^3, y_{78}) \tag{14}$$

$$\neg z_8^1 \vee AtMostOne(y_{s8}^2, y_{s8}^3, y_{78}) \tag{15}$$

という制約をもつ（C 社も同様）．

制約 6 共有運用について，自社の編成を割り当てないとき，そこから出る運用パスは 0 本である．

例えば，図 1 の共有運用 o_7 について，A 社は

$$(z_7^1 \vee \neg y_{7_t}^1) \wedge (z_7^1 \vee \neg y_{7_t}^2) \wedge (z_7^1 \vee \neg y_{78}) \tag{16}$$

という制約をもつ（C 社も同様）．これは，A 社が運用 o_7 に自社の編成を割り当てない場合，o_7 から出る有向リンクはどれも運用パスには含まれないことを意味する．

2.3.3 最終日ノードに関する制約

最終日ノード t_i に関する制約は次の 1 つのみである．

制約 7 各編成 h_i の最終日ノード t_i に入る h_i の運用パスはちょうど 1 本である．

例えば，図 1 の A 社の最終日ノード t_2 へは，それぞれノード s_2, o_2, o_7 から 3 本の有向アークが入るが，h_2 の運用パスに含まれるのは 1 本だけなので，

$$AtLeastOne(y_2, y_{2t}^2, y_{7t}^2) \tag{17}$$

$$AtMostOne(y_2, y_{2t}^2, y_{7t}^2) \tag{18}$$

という制約を A 社は所有する．

以上のように，各社が制約 1 から 7 を SAT の節として記述して節集合分割型分散 SAT を構成し，分散アルゴリズムである M-ABTS [下牧 12] で解く．なお，M-ABTS で各社（エージェント）がメッセージ通信により情報交換するのは，共有運用に関する変数 z_j^k （共有運用 o_j で会社番号 k の編成を利用するとき真，そうでないとき偽）の値のみである．

表 1: 集中解法による実行時間 [sec]

問題例	変数	節	解	MiniSAT	glucose	CPLEX
han+kin-h32-o53	9534	27341	UNSAT	T.O.	T.O.	**1.50**
han+kin-h36-o53	11306	32709	UNSAT	T.O.	1523.80	**1.90**
han+kin-h37-o53	10800	33666	SAT	2.25	**0.62**	1.74
han+kin-h40-o53	16002	47046	SAT	4.04	**1.84**	2.43
han-h55-o85	43620	180818	SAT	6.46	**6.25**	20.55
tkdshin-h80-o81	84458	470549	SAT	157.80	**18.56**	112.73
kin-h50-o137	48012	266286	SAT	**7.95**	28.30	226.34
kin-h55-o172	61798	392727	SAT	69.15	**15.88**	1038.80
han-h85-o202	120249	718179	SAT	606.82	**161.33**	1549.14
tkdshin-h110-o258	231371	2663259	SAT	T.O.	**1208.39**	T.O.

表 2: 分散解法による実行時間 [sec]

問題例	変数	共有変数	節	M-ABTS
han+kin-h37-o53-1	5344	113	38100	**7.612**
han+kin-h37-o53-2	5307	112	37772	**6.939**
han+kin-h40-o53	7098	250	47672	**15.406**
han+kin+san-h56-o80	7182	148	52957	**19.479**
han+kin+san-h59-o80	7652	154	57032	**8.462**
han+kin+san-h93-o161	21555	282	198305	T.O.
han+kin+san-h96-o161-1	22442	288	207245	T.O.
han+kin+san-h96-o161-2	21975	288	205155	**2232.373**
han+kin+san-h100-o161-1	24057	296	226906	T.O.
han+kin+san-h100-o161-2	24130	296	230807	T.O.

3. 実験

実際の鉄道ダイヤを参考に問題例を作成し，自社運用と共有運用を区別せずに全体を 1 つの問題として解く集中解法，および，自社運用と共有運用を区別して複数の鉄道会社が協調的に全体の問題を解く分散解法の性能を実験で評価する．

問題例は阪神本線，山陽本線，近鉄奈良線，東海道新幹線のダイヤから作成した．なお，実際の車両運用計画のデータは公表されていないため，各編成の出入庫場所を仮想的な車庫とし，適宜，車庫からダイヤの初期出発場所，ダイヤの最終日到着場所から車庫に編成を移動させる擬似運用を追加した．

集中解法としては，SAT に定式化し，SAT ソルバーとして MiniSAT 2.2.0 [Eén 03]，glucose 2.1 [Audemard 09] を適用する方法，および，0-1 整数計画問題（多品種フロー問題）に定式化し，IBM CPLEX 12.3.0 を適用する方法を用いた．また，分散解法としては M-ABTS [下牧 12] を利用する．なお，M-ABTS で各エージェントが局所問題を解くのに利用する SAT ソルバーは MiniSAT 2.2.0 である．

実験環境は，集中解法および分散解法ともに CPU: Intel Core i7-870@2.93GHz（4 コア 8 スレッド）, Memory: 8GB, OS: Ubuntu 11.04, Compiler: Java 1.60_22 である．分散解法は当該マシン上に離散事象シミュレータを作成し，その上に実装した．

表 1 は集中解法による実行時間である．なお，実行の制限時間をそれぞれ 1 時間とし，それを超えた場合は T.O.（Time Out）と表記する．問題例のラベルは，左から「路線名-h 編成数-o 運用数」であり，路線名は”han”が阪神本線，”san”が山陽本線，”kin”が近鉄奈良線，”tkdshin”が東海道新幹線で，共有運用を含む問題例は関連する路線名を”+”でつないで表記する．また，各問題例について，それを SAT に定式化した場合の変数と節の数を明記した．表 1 の最初の 2 例は，運用数に対して編成数が十分でない充足不可能（UNSAT）な問題例である．今回の実験では，そのような問題例に対しては CPLEX が優れていることがわかる．一方で，充足可能（SAT）である他の 8 問については SAT ソルバー（特に glucose）が優れていることがわかる．

表 2 は分散解法による実行時間である．表 2 の最初の 3 例は阪神本線と近鉄奈良線の共有運用を含む 2 社間の分散 SAT，他の 7 例は阪神本線と近鉄奈良線，阪神本線と山陽本線のそれぞれの共有運用を含む 3 社間の分散 SAT の例である．なお，分散解法（M-ABTS）の実行時間とは，エージェントが送信する各メッセージについて，それが送信されるまでに費やされた累積実行時間を記録し，解法が終了した時点での全メッセージの累積実行時間の最大値である．今回の実験では，3 社間の分散 SAT の 7 例中 4 例が制限時間内に解けていないことがわかる．

4. まとめ

本稿では，車両運用計画問題を SAT として定式化し，SAT ソルバーを用いて解く新たな解法を提案した．また，複数の鉄道会社が運用を一部共有して相互直通運転を行なう際の車両運用計画問題を節集合分割型分散 SAT で定式化し，分散解法である M-ABTS で解くことを提案した．

今後の課題は分散解法の性能改善である．現在の M-ABTS では，すべてのエージェントが SAT ソルバーで局所問題を解いているが，局所問題が UNSAT のときは一般に CPLEX の方が効率的だと予想される．よって，両者を併用する分散解法に今後取り組む予定である．

参考文献

[Audemard 09] Audemard, G. and Simon, L.: Predicting Learnt Clauses Quality in Modern SAT Solver, in *Proceedings of 21st International Joint Conference on Artificial Intelligence (IJCAI-2009)*, pp. 399–404 (2009)

[Cordeau 00] Cordeau, J.-F., Soumis, F., Desrosiers, J., and Desrosiers, J.: A Benders Decomposition Approach for the Locomotive and Car Assignment Problem, *Transportation Science*, pp. 133–149 (2000)

[Eén 03] Eén, N. and Sörensson, N.: An Extensible SAT-Solver, in *Proceedings of 6th International Conference on Theory and Applications of Satisfiability Testing (SAT-2003)*, pp. 502–518 (2003)

[今泉 10] 今泉 淳, 山岸 雄樹, 森戸 晋：二段階数理計画アプローチによる鉄道車両運用計画の策定, 日本オペレーションズ・リサーチ学会和文論文誌, Vol. 53, pp. 14–29 (2010)

[井上 10] 井上 克巳, 田村 直之：特集 最近の SAT 技術の発展, 人工知能学会誌, Vol. 25, No. 1, pp. 56–129 (2010)

[大槻 10] 大槻 知史, 愛須 英之, 田中 俊明：車両運用計画問題に対する制約充足解法の提案, 日本オペレーションズ・リサーチ学会和文論文誌, Vol. 53, pp. 30–55 (2010)

[下牧 12] 下牧 昌太郎, 平山 勝敏：節集合分割型分散 SAT に対する非同期バックトラッキングアルゴリズム, 2012 年度人工知能学会全国大会（第 26 回）論文集 (2012)

[Sinz 05] Sinz, C.: Towards an optimal CNF encoding of boolean cardinality constraints, In Proc. of the 11th Intl. Conf. on Principles and Practice of Constraint Programming （CP 2005）, pp. 827–831 (2005)