修士学位論文

時間的資源制約条件下における充足可能性問題に対するリスタートスケジューリングの効果

(Restart Strategies for Time-Constrained SAT Solvers)

平成 26 年度

広域科学専攻 · 広域システム科学系

31-136804

猪鼻　真裕

目次

1 序論

1.1 背景

一般に，ある問題を解く場合には複数の解法が考えられる．例えば，充足可能性問題 (SAT) やプランニング問題を解くためには様々なアルゴリズムが開発され，その性能を競っている [3][15]．各アルゴリズムの性能は問題インスタンスによってばらつき，特定の 1 つのアルゴリズムがどの問題インスタンスにおいても優れているということはほとんどない．No Free Lunch 定理 [28] は，あらゆる全問題についてこのことを述べた定理[*1] であるが，実際には充足可能性問題 (SAT) などに問題インスタンスの範囲を狭めても，各ソルバの性能は問題毎に異なる．

同様のことは，乱数を用いる 1 つのアルゴリズムにおいてもいえる．同じアルゴリズムであっても，擬似乱数生成のシードに依存して結果にばらつきが生じる．Huberman ら [14] は，この問題に対して複数のアルゴリズムを組み合わせるポートフォリオ戦略を提案した．ここでは，異なる乱数シードを用いる場合も複数のアルゴリズムとみなしている．複数のアルゴリズムを組み合わせることによって，実行時間の期待値を小さくするとともに，各実行における実行時間のばらつきも小さくできることに注目している．ポートフォリオ戦略は性能もよく，ロバストであることが，プランニング問題 [13][27]，充足可能性問題 (SAT)[12][29]，遺伝的アルゴリズム [22][11] など，様々な分野で研究されている．

一方，同様の観点から，ある確率的なアルゴリズムを一定時間だけ実行したら別の乱数シードを用いて探索をやり直す手法が，リスタート戦略として研究されてきた [17]．ポートフォリオ戦略では時間資源を複数のアルゴリズムに分配するのに対し，リスタート戦略は一定時間は同一アルゴリズムを実行し続ける，という違いはあるが，リスクヘッジという観点では両者はどちらも良い効果があると考えられている．リスタート戦略も，SAT[10] や遺伝的アルゴリズム [8] などで多くの研究がなされている．

ポートフォリオやリスタートスケジュールを作成する際には，どれだけの時間をどのアルゴリズムに分配するか，という問題がある．最も単純な考え方は，各アルゴリズムに均等に資源を分配する方法である．Luby ら [17] は，ある乱数を用いる確率的アルゴリズムがある問題インスタンスを解くときの実行時間の確率分布を完全に知っている場合には，

[*1] 厳密には，(i) 離散的かつ有限な探索空間及び評価値で，(ii) 性能をアルゴリズムが返す評価値で測り，(iii) 探索箇所は重複せず，(iv) 問題に対する事前知識は仮定しない，という条件下では他のどのアルゴリズムよりも平均性能が良いアルゴリズムは存在しない，という定理．

この均等な時間でリスタートするスケジュールが最適なスケジュールであることを理論的に証明した.

しかし，解きたい問題の実行時間の確率分布を事前に知っているという仮定は，実用的には成り立たない．そこで，手元にある別の問題を解いた結果から，よいポートフォリオやリスタートスケジュールを学習する方法が研究されている [29][8]．SAT においては，Streeter ら [26] が，手元のデータベースから最適な 1 つのリスタートスケジュールを生成する手法を提案した．しかし，解きたい問題毎に最適なリスタートスケジュールは異なるはずである．また，Streeter ら [26] の場合には，ある特定のドメインのみしか扱われておらず，リスタートスケジュールを学習から求める手法については，十分に研究されているとはいえない.

1.2 本研究の目的

そこで本研究では，遺伝的アルゴリズムに用いられた，過去の実行記録のデータベース (Past performance Data Base; PDB) からサンプリング法を用いてリスタートスケジュールを生成する手法 [8] を，SAT に適用することを考える.

リスタートスケジュールの効果を測るためには探索途中の効用を定義する必要があるが，SAT ソルバの場合，探索途中の状態を評価するのは困難である．本研究では，時間的資源制約条件下での問題設定を行うことで，その問題を解決する．そして，この問題設定のもとではリスタート戦略は有効かどうかを検討する.

また，PDB からスケジュール推定に用いる学習データベースを問題毎に選択し，作成する方法について調査する．手元のデータから，どのデータを学習に用いるかの選択は学習の結果に大きな影響を与えると考えられるが，これまで十分に研究されているとはいえない．本研究では，既存研究で用いられている作成方法に加えて，リスタートが有効な問題，という作成方法を提案する.

最後に，リスタート戦略の有効性について，SAT の問題ファイルから計算される特徴ベクトルから推測可能かどうかを検証する．時間制約がなく，時間資源に無限大を仮定できる場合はリスタート戦略を用いて実行時間の期待値や分散を低くすることは可能である．しかし，時間制約がある場合，探索途中でリスタートすることは時間制約以内に解き終わらないリスクを伴う．リスタート戦略が有効でない場合は，リスタート戦略は行わない方が望ましい．逆にリスタートが有効な場合には，リスタートを行うことで時間制約以内に解き終わる確率を上げることができる．そのため，リスタート戦略の有効性をアルゴリズムの実行前に判定できることは重要な意味を持つ.

本論文は，以下の通りに構成される．まず 2 章では，充足可能性問題と関連する研究について述べ，本研究の位置付けを確認する．次に 3 章で時間的資源制約の問題設定を述べ，効用関数を定義する．また，本研究で用いるアルゴリズムについても説明する．4 章では，ある問題をあるアルゴリズムで解く場合の実行時間の確率分布を知っているという仮定のもとで，リスタート戦略の有効性について検討する．5 章では，リスタート戦略が有効である問題について，PDB を用いてリスタートスケジュールを推定する方法が有効であるかどうかを検討する．また，学習データベースの作成方法についても比較する．6 章でリスタート戦略の有効性を事前に判定可能であるかどうかを検討し，7 章に本研究の結論と今後の課題をまとめる．

2　充足可能性問題と関連研究

本章では，まず充足可能性問題 (SAT) の概要について説明する．そして SAT を解くアルゴリズムである SAT ソルバについて，特に SAT ソルバにおけるポートフォリオ戦略と，リスタート戦略について概観し，本研究の位置付けを述べる．最後に，遺伝的アルゴリズム (GA) に用いられ，本研究で扱うリスタート戦略について説明する．

2.1　充足可能性問題

人工知能の分野における問題の多くはブール式または命題論理式で定式化され，その充足可能性の判定や充足割り当ての発見について多くのアルゴリズムが研究されている．また，ハードウェア検証への応用 [2] などもあり，工業的な分野でも SAT での定式化と SAT ソルバを用いた検証の有効性が認められている．

本節では，制約充足問題やプランニング，自動推論 · 定理証明など様々な問題の基本となる，充足可能性問題 (SAT) ついて説明する[*2]．

まず，各変数が $\{0,\ 1\}$ の値をとるブール変数の集合 X を考える．変数 $x_i \in X$ の否定を $\neg x_i$ と書き，$\neg x_i = 1 - x_i$ とする．各変数 $x_i \in X$ 及びその否定 $\neg x_i$ をリテラルと呼び，前者を正リテラル，後者を負リテラルと呼ぶ．これらのリテラル $l_i \in \{x_1,\ x_2,\ \ldots,\ x_n,\ \neg x_1,\ \neg x_2,\ \ldots,\ \neg x_n\}$ を用いて，

$$l_1 \ \vee \ l_2 \ \vee \ \cdots \ \vee \ l_m$$

と書ける論理和 (選言) を節 (clause; クローズ) と呼ぶ．節に含まれるリテラルの個数をその節の長さという．例えば，節 $C = (l_1 \ \vee \ l_2 \ \vee \ \cdots \ \vee \ l_m)$ の長さは m である．長さが 1 の節を単位節 (unit clause) と呼ぶ．

次に，変数への値の割り当てを考える．各変数 $x_i \in X$ に対して 0 または 1 の真偽値を割り当てることを関数 $\nu : X \to \{0,\ 1\}$ で表し，真偽割当てと呼ぶ．$|X| = n$ のとき，真偽割当ての数は 2^n 通り存在する．ある真偽割当て ν により，節 C の論理値が 1 となるとき，ν は C を充足する (satisfy) という．節の論理積 (連言) からなる命題論理式

$$\psi = C_1 \ \wedge \ C_2 \ \wedge \ \cdots \ \wedge \ C_k$$

を連言標準形 (conjunctive normal form; CNF) という．CNF 形式の命題論理式 ψ があり，真偽割当て ν のもとで ψ を評価すると 1 となるとき，ν は節の集合 $\{C_1,\ C_2,\ \ldots,\ C_k\}$ また

[*2] [23][16][31] などを参考にまとめた．

は ψ を充足するといい，ν を充足割当て (satisfying assignment) と呼ぶ．命題論理式 ψ を充足する充足割当て ν が存在するとき，すなわち，ψ 中の全ての節 C_i において，少なくとも 1 つのリテラルが 1 となる ν が存在するとき，ψ は充足可能 (satisfiable) であるという．例えば，以下の命題論理式

$$(x_1 \lor \neg x_2) \land (\neg x_1 \lor \neg x_3) \land (x_2 \lor \neg x_3)$$

において，全ての変数に 1 を割り当てる真偽割当て ν は，二番目の節を 0 にするため充足割当てではない．逆に，全ての変数に 0 を割り当てる真偽割当て ν' は，充足割当てである．命題論理式 ψ を充足する真偽割当てが存在しないとき，ψ は充足不能 (unsatisfiable) であるという．例えば，$x_1 \land \neg x_1$ は充足不能な式である．

以上の定義を用いて，ある命題論理式 ψ が与えられたとき，ψ が充足可能 (satisfiable) か充足不能 (unsatisfiable) かを判定する問題を，充足可能性問題 (Satisfiability Problem; SAT) と定義する．任意の命題論理式は，論理的同値である CNF 式に指数時間オーダーで変換可能であり，充足可能性が同値である CNF 式に線形時間オーダーで変換可能であるため，SAT を CNF 形式に限定しても一般性を失わない．CNF 形式の命題論理式 ψ において，ψ 中の全ての節の長さが k のとき，ψ の充足可能性問題を k-SAT と呼ぶ．

SAT は初めて NP 完全問題として証明された問題である [4]．厳密には，$k \geq 3$ の場合に k-SAT が NP 完全となる．任意の命題論理式における SAT は，3-SAT に多項式時間で変換可能である．2-SAT には多項式時間のアルゴリズムが存在する．また，高々 1 つの正リテラルしか含まないような節，すなわち

$$C = \neg x_1 \lor \neg x_2 \lor \cdots \lor \neg x_{n-1} \lor x_n$$

の形をした節をホーン節と呼ぶ．CNF 形式の命題論理式 ψ 中の節が全てホーン節のとき，ホーン SAT (HORNSAT) とよぶ．ホーン SAT にも，多項式時間のアルゴリズムが存在する．ホーン節 C は，

$$x_1 \land x_2 \land \cdots \land x_{n-1} \rightarrow x_n$$

という命題論理式と同値である．これはモーダスポーネンスの標準形であり，一階述語論理において重要な意味を持つ．

2.2 SAT におけるポートフォリオ

SAT は，一般には NP 困難な問題であり，多項式時間で解けるアルゴリズムは開発されていない．SAT を解く既存の SAT ソルバは，系統的で完全なアルゴリズムと，局所探索を

用いた不完全なアルゴリズムの 2 つに大別できる [31]．どちらのアルゴリズムも，探索において乱数を用いるものが主流であるため，SAT ソルバは確率的なアルゴリズムと考えてよい．

Huberman ら [14] は，経済学におけるポートフォリオの概念を援用し，確率的なアルゴリズムにおいては複数のアルゴリズムに時間資源を分配することで，平均的な実行時間を短くし，かつ実行時間のばらつきを抑えることが可能であることを示した．

その後ポートフォリオと呼ばれる戦略は，様々な種類のものが研究されている．ここでは，資源分配逐次型，資源分配並列型，アルゴリズム選択型，アルゴリズムスケジュール型の 4 種類に分類し，それぞれ説明する．

資源分配逐次型は，Huberman ら [14] が用いたもので，逐次プロセッサ上で並列的に資源分配する枠組みである．イメージ図を図 1(a) に示す．

資源分配並列型は，複数のアルゴリズムを並列に実行する枠組みである．ManySAT[12] は，リスタート戦略，変数選択ヒューリスティック，真偽割当て選択ヒューリスティック，節の学習法において，複数の異なる手法を組み合わせた異なるアルゴリズムを 4 つ用意し，並列実行した．n コアに並列化する場合，逐次プロセッサ上での資源分配法に比べて使用可能な時間資源は n 倍になる．並列化を行う場合は，コア数を n 倍にしたときに性能は何倍になるかが，重要な問題となる．

アルゴリズム選択型は，複数のアルゴリズムの集合からなるアルゴリズムプールを用意し，解きたい問題に対し，最も性能がよいと期待されるアルゴリズムを 1 つ選び，実行する枠組みである．SATZilla[29] はこの枠組みを用い，state-of-the-art の性能を示した．

アルゴリズムスケジュール型は，あるアルゴリズム A_1 を一定時間 t_1 だけ実行した後，別のアルゴリズム A_2 を一定時間 t_2 だけ実行する，という手順を繰り返すときのスケジュール $S = \{ (A_1, t_1), (A_2, t_2), \dots \}$ を生成する枠組みである．Streeter[27] らは，最適なアルゴリズムスケジュールの 4 近似スケジュールを生成する多項式時間のアルゴリズムを提案し，4 近似未満のスケジュール生成は NP 困難であることを示した．

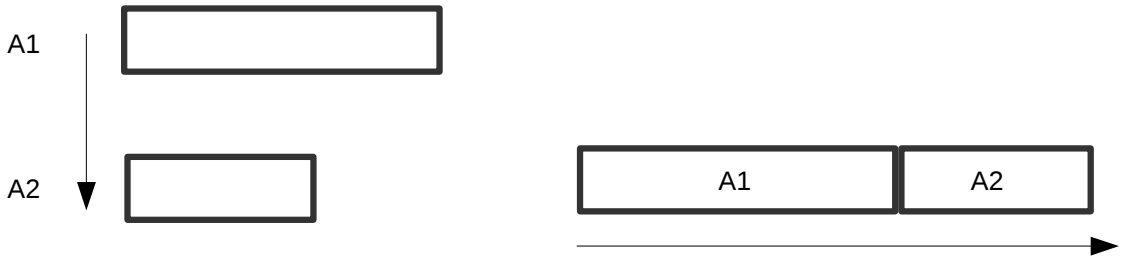
これらのポートフォリオ戦略は，複数の全く異なるアルゴリズムをいかにうまく組み合わせるかに重点をおいた研究である．本研究では，ある 1 つのアルゴリズムに注目し，その 1 つのアルゴリズムの性能改善に重点をおき，次節で述べるリスタート戦略を用いる．

2.3 SAT におけるリスタート戦略

ポートフォリオ戦略が一般に複数のアルゴリズムを想定しているのに対し，リスタート戦略は特に 1 つのアルゴリズムとその実行時間の確率分布に注目し，実行時間の期待値を

どう改善するかに重点をおいている．SAT や組合せ最適化問題などを解く完全なソルバにおいては，探索初期における変数選択の違いにより実行時間が大きく変動する，Heavy-tail 現象が報告されている [9]．Gomes ら [10] は，Heavy-tail 現象を示す完全な SAT ソルバに対し，リスタート戦略が有効であることを示した．

ただしここで，異なる乱数シードを用いて実行される確率的アルゴリズムは，異なるアルゴリズムだと考える場合，リスタート戦略は複数のアルゴリズムへの時間資源の分配ともいえ，ポートフォリオ戦略の一種といえる．特に，資源分配逐次型とは類似し，アルゴリズムスケジュール型とは一致する．資源分配逐次型ポートフォリオ戦略とリスタート戦略の違いのイメージ図を，図 2.3 に示す．資源分配逐次型ポートフォリオ戦略では並列的に資源を分配するのに対し，リスタート戦略では，ある 1 つのアルゴリズムを一定時間実行し続けた後に，別のアルゴリズムを実行する．



(a) 資源分配逐次型ポートフォリオ戦略　　(b) リスタート戦略

図 2.1　資源分配型ポートフォリオ戦略とリスタート戦略の違い

Luby[17] は，あるアルゴリズムがある問題を解くときの実行時間の確率分布が既知の場合かつ，実行時間に制限がない場合 (無限大まで確率分布が定義される場合) には，ある値 t^* が存在し，その時間が経過したら常にリスタートするスケジュール $S_{t^*} = \{t^*, t^*, \ldots, \}$ が実行時間の期待値を最小化するスケジュールであることを証明した．

また，実行時間の確率分布を全く知らない場合に，最適なリスタートスケジュールを用いた場合の実行時間の期待値を l^* とするとき，$O(l^* \log l^*)$ の実行時間の期待値を持つ普遍的スケジュールを提案した．これは Luby スケジュールとして知られ，他の多くの SAT ソ

ルバ [7][12] に用いられている．

Streeter ら [26] は，手元にあるデータベースからよりよい普遍的スケジュールを計算する手法を提案し，理論的に解析した．すなわち，ある問題群とそれらの実行時間の確率分布が既知の場合，全問題を解く場合の実行時間の期待値の最小値から，4 倍以内の実行時間で解き終わるリスタートスケジュールを求める多項式アルゴリズムを提案した．また，そのスケジュール生成法を，あるドメイン内の問題群の列 $\langle x_1, x_2, \ldots, x_n \rangle$ を対象とし，それらを逐次的に解くオンラインの状況に応用した．しかしこの研究では，特定ドメインの問題をランダムに生成する方法で問題群が生成されており，十分に一般的とは言えない．また，普遍的スケジュールの生成は意味がある一方で，各問題における効果的なリスタートスケジュールは，それぞれ異なることが予想される．本研究では，後述するように様々なドメインの問題群から，解きたい問題毎に学習を行う，より一般的な状況を考える．

適応的にリスタートする区間を調節する手法も研究されているが [1][24]，適応のためには複数のパラメータが存在し，それらをうまく調節しないとパフォーマンスがうまくいかないという問題点がある．

2.4 過去の実行記録のデータベースを用いる方法

Fukunaga[8] は，過去の実行記録のデータベース (Past performance Data Base; PDB) を用いて問題毎にリスタートスケジュールを生成する手法を提案し，遺伝的アルゴリズム (Genetic Algorithm; GA) に適用した．

遺伝的アルゴリズムは，ある目的関数 $h(z)$ を最大化，または最小化する解 z を近似的に求めるアルゴリズムであり，時間ステップ t 毎にある評価値 H_t が計算される．その時間推移を PDB として記録しておき，リスタートスケジュールの評価に用いる．すなわち，アルゴリズム A が問題インスタンス x を時間 t で解く場合の効用を t ステップ目の効用 H_t を用いて，

$$U(A,\ x,\ t) = H_t$$

と定義する．そして，あるリスタートスケジュール $S = \{t_1, t_2, \ldots, t_n\}$, $\sum_{i=1}^{n} t_i \leq T$ を適用した場合の効用を

$$U(A,\ x,\ T,\ S) = max(U(A,\ x,\ t_1),\ U(A,\ x,\ t_2),\ \ldots,\ U(A,\ x,\ t_n))$$

と定義する．

未知の問題インスタンス x を解く際に，別の問題インスタンス群を解いた結果の PDB を用いて有効なリスタートスケジュールを生成し，適用する．PDB は，アルゴリ

ズム A[*3]，問題インスタンス x，乱数シード $seed$ の組 $(A,\ x,\ seed)$ と，その設定下における探索履歴からなる．探索履歴は，時間ステップとその時刻での評価値の組の集合 $\{(t,\ H_t),\ (2t,\ H_{2t}),\ (3t,\ H_{3t}),\ \ldots\}$ で定義する．記録する探索ステップの幅を 10 とし，最大化問題を GA で解く場合の PDB の簡単な例を，以下に示す．

GA の PDB

$$data(A_1,\ x_1,\ seed_0) = \{(10,\ 1),\ (20,\ 2),\ (30,\ 2),\ (40,\ 3)\}$$

$$data(A_1,\ x_1,\ seed_1) = \{(10,\ 1),\ (20,\ 1),\ (30,\ 2),\ (40,\ 3)\}$$

$$data(A_1,\ x_2,\ seed_2) = \{(10,\ 1),\ (20,\ 1),\ (30,\ 2),\ (40,\ 2)\}$$

$$data(A_1,\ x_2,\ seed_3) = \{(10,\ 1),\ (20,\ 1),\ (30,\ 1),\ (40,\ 2)\}$$

$$\vdots$$

$$data(A_l,\ x_m,\ seed_n) = \{(10,\ 1),\ (20,\ 2),\ (30,\ 3),\ (40,\ 4)\}$$

このように探索履歴を記録する PDB が作れる場合，リスタートスケジュールの効用の定義及び効用の推定は自然に可能である．また，効用の推定に後述するサンプリング法を用いることで，問題毎に現実的な時間で有効なリスタートスケジュールを計算することが可能である．

次章では，この PDB を用いて有効なリスタートスケジュールを生成する手法を SAT に適用する方法について説明する．

[*3] あるアルゴリズムにおいて，異なるパラメータ設定は異なるアルゴリズムと考える．すなわち，アルゴリズム A と表記する場合，様々なパラメータ値は特定の値に設定されていることを想定する．

3　時間的資源制約条件

本章では，前章で言及した Past performance Data Base を用いてリスタートスケジュールを生成する方法を，充足可能性問題 (SAT) に適用することを考える．まずは SAT における PDB の構成要素について説明し，ある時間 t における効用を時間的資源制約条件の下で定義する．その結果を用いて，スケジュール生成の際の探索空間のサイズについて議論する．最後に，本研究で用いるアルゴリズムについて述べる．

3.1　SAT における PDB

GA における PDB の構成要素は，アルゴリズム A，問題インスタンス x，乱数シード $seed$ 及び，探索過程の履歴であった．前者の 3 つについては，SAT ソルバの場合にもそのまま当てはめることができる．しかし探索履歴については，適応的リスタート戦略の指標として用いられているもの [24] はあるが，判定が終了する確率，すなわち本研究における効用との関係は明らかでない．そこで本研究では，判定が終了した時刻 t_{solved} しか得られないと仮定し，PDB を作成する．SAT ソルバの探索過程の評価値の設定とその利用は，今後の課題である．

以上の仮定の下で，SAT の場合の PDB の簡単な例を以下に示す．

SAT の PDB

$$data(A_1,\ x_1,\ seed_0) = \{\ t_{1solved}\ \}$$
$$data(A_1,\ x_1,\ seed_1) = \{\ t_{2solved}\ \}$$
$$data(A_1,\ x_2,\ seed_2) = \{\ t_{3solved}\ \}$$
$$data(A_1,\ x_2,\ seed_3) = \{\ t_{4solved}\ \}$$
$$\vdots$$
$$data(A_l,\ x_m,\ seed_n) = \{\ t_{nsolved}\ \}$$

仮に，判定が終了した時刻 t_{solved} における効用を 1，それ以外の時刻の効用を 0 にすると，PDB 内の多くの部分で時刻 t の効用 $U(A,\ x,\ t)$ が 0 になってしまう．

$U(A,\ x,\ t)$ が計算できないと，リスタートスケジュール S の効用 $U(S,\ x,\ t,\ S)$ も計算できない．この問題を，次節に示す時間的資源制約条件を加えることで解決する．

3.2 時間的資源制約条件下における問題設定

前節で述べた問題点を解決するために，時間的資源制約を考える．すなわち，ある時刻 T を設定し，T 以内ならいつ判定が終了しても構わない，とし，T 以内に終了する確率を最大化することを考える．これは実問題では，例えば，ジョブスケジューラにジョブを投げる際に，適当な walltime を指定し，その walltime 以内に解き終わる確率を最大化させたい場合などに対応する．

この設定の下で，効用 $U(A,\ x,\ T)$ を

$$U(A,\ x,\ T) = \mathrm{Prob}[x \text{ is solved by } A \text{ within time limit } T]$$

と定義する．これは，アルゴリズム A の挙動が乱数シード *seed* によって確率的に変動することを反映している．また，$p(t)$ を A が x の判定を終了する実行時間の確率分布とし，$q(t) = \int_0^t p(t')dt'$ なる累積確率分布を用いると，

$$U(A,\ x,\ T) = q(T)$$

と書ける．

効用 $U(A,\ x,\ T)$ を最大化することは，T 以内に解き終わる確率を最大化することに対応する．これを，本研究の問題設定とする．

3.3 リスタートスケジュールの効用とスケジュール生成のための探索空間

ある時刻 t における効用 $U(A,\ x,\ t)$ が定義できたため，それを用いてリスタートスケジュール S の効用を定義する．問題設定が，時刻 T 以内に解き終わる確率を最大化することだったため，リスタートスケジュール S を適用した場合の効用は，各リスタートのいずれかの区間において少なくとも 1 回判定が終了する確率と定義する．すなわち，$S = \{t_1,\ t_2,\ \ldots,\ t_n\},\ \sum_{i=1}^{n} t_i \leq T$ として，

$$\begin{aligned} U(A,\ x,\ T,\ S) = &\quad 1 - \prod_{i=1}^{n}(1 - q(t_i)) \\ =&1 - \prod_{i=1}^{n}(1 - U(A,\ x,\ t_i)) \end{aligned}$$

とする．

ここで，この効用の定義の下では，リスタート区間の順序について対称性があることに注意する．例えば，$S = \{t_1,\ t_2,\ t_3\}$ というスケジュールと $S = \{t_2,\ t_1,\ t_3\}$ というスケジュー

ルは，積の可換性より同じ効用を持つ．ただし，これは各リスタートが独立な場合に成り立つ．

これは，既存研究の，リスタートスケジュールの効用を判定が終了する時間の期待値で測る場合とは大きく異なる．効用を判定が終了する時間の期待値で測る場合は，リスタートスケジュールの順序は決定的に重要なものとなる．例えば，$S = \{t_1, t_2, t_3\}$ と $S = \{t_2, t_1, t_3\}$ では，$t_1 \neq t_2$ の場合，異なる効用を持つ．すなわちここでは，$S = \{t_1, t_2, \ldots, t_n\}$ は列である．

時間的資源制約条件の問題設定の場合，$S = \{t_1, t_2, \ldots, t_n\}$ は列ではなく，重複を許す集合である．この結果，解の候補となるスケジュールを探索する際に，$S = \{t_1, t_2, \ldots, t_n\}$ の順序を無視して考えてよい．すなわち，探索空間を $n!$ 通り削減することが可能となる．スターリングの公式を用いると，$O(n!) \approx O(n^n)$ となるため，探索空間を指数的に削減することが可能となる．

また，この結果は，SAT ソルバに関わらず，アルゴリズム A が定まった解を返すアルゴリズム一般について成り立つ．例えば，制約充足問題を解くソルバや，最適解を求めるプランニング問題を解くプランナなどについても適用可能である．

3.4 本研究で用いるアルゴリズムの概要

本節では，本研究で用いるアルゴリズムについて具体的に述べる．PDB を用いてリスタートスケジュールを推定するアルゴリズムは，アルゴリズム 1 に示すように，大きく分けて 2 つの部分からなる．

Algorithm 1 Estimate_Restart_Schedule(PDB, $NumSamps$, T, k)

$trainingDB \leftarrow$ Extract_trainingDB(PDB)

$schedule* \leftarrow$ Find_best_schedule($trainingDB$, $NumSamps$, T, k)

1 つ目は，複数の問題を，複数のアルゴリズムと複数の乱数を用いて解いた結果を集めた PDB から，スケジュール推定に用いるデータを取り出す部分である．2 つ目は，選んだデータの集合から最適だと考えられるスケジュールを計算する部分である．以下，それぞれについて説明する．

3.4.1 学習データベースの選び方

一般に手元には，様々な問題を，様々なアルゴリズムや乱数シードを用いて解いた結果がある．ある 1 つの問題を解く場合，それらのデータを全部用いて学習するのでは，うま

く学習できないことが考えられる．そこで本研究では，学習データベースをどのように選べば有効なリスタートスケジュールが生成できるかを，実験によって調べる．

ただし，アルゴリズムについては，(i) 1 つのアルゴリズムの性能を改善するというリスタート戦略の目的と，(ii) あるアルゴリズムのリスタートスケジュールは同じアルゴリズムの結果から学習するのがよいという仮定から，1 つのアルゴリズムを用いた結果のみを PDB に用いる．PDB や学習データベースを複数の異なるアルゴリズムで構成する場合のリスタートスケジュールの生成については，今後の課題である．詳細は，4.1 節にて述べる．

よって，複数の問題，複数の乱数からなる PDB を用意し，その中のデータを用いて学習データベースを作成する．

具体的には，以下の選び方について調査する．

- 全問題を学習データベースとして選択する場合
- 同じ問題カテゴリを学習データベースとして選択する場合
- 同じ問題ドメインを学習データベースとして選択する場合
- リスタートが有効な問題群を学習データベースとして選択する場合

より詳しい説明は，4.1 節にて述べる．実験結果は，5.7 節に示す．

3.4.2 最適スケジュールの計算

学習データベースが決まったら，それを用いて最適スケジュールを計算する．最適スケジュールを計算する手順の擬似コードを，アルゴリズム 2 に示す．

まずはリスタートスケジュールの候補を複数生成する．生成されるスケジュール候補の詳細については，3.4.3 節に示す．生成された候補内の各スケジュールについて，学習データベースを用いて効用を計算する．そこで，最も効用が高いスケジュールを返す．このとき，学習データベース内のデータによって計算されるスケジュールの効用は異なる．そのため，何を学習データベースに選ぶかが，どんなリスタートスケジュールを適用するかに効いてくる．

3.4.3 スケジュール候補の生成

Fukunaga[8] は，時間資源 T を k 秒単位に分割し，あり得る組み合わせを全て列挙し，スケジュール候補を作成した．時間資源の制約を表す T に対し，k はスケジュールの精度を表す．例えば，$T = 300$, $k = 100$ のとき，あり得るスケジュール候補は $\mathcal{S}$ = { {300}, {200, 100}, {100, 100, 100} } の 3 通りである．生成されるスケジュール候

Algorithm 2 Find_best_schedule(*trainingDB*, *NumSamps*, T, k)

```
bestSchedule = {}
bestUtility = 0
S = Generate_schedules(T, k)
for each S ∈ S do
    utility ← Estimate_utility(S, NumSamp, trainingDB)
    if bestUtility < utility then
        bestSchedule ← S
        bestUtility ← utility
    end if
end for
Return bestSchedule
```

補数は T/k をパラメータとして指数的に増大する．その様子を図 3.1 に示す．

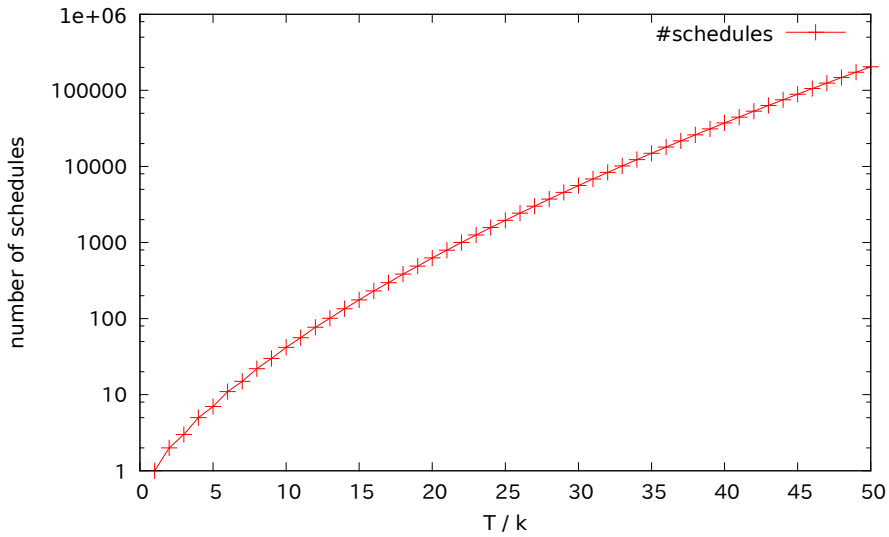


図 3.1　T/k とスケジュール候補数の関係

このとき，3.3 節で述べた S の性質が重要となる．すなわち，この問題設定において S は列ではなく重複を許す集合であるため，全列挙に要する計算が実用的な時間で済む．本研究では，これを全列挙法として使用する．

上記の全列挙法は，k を調節することで十分高い精度で最適なリスタートスケジュールを生成可能である．しかし，最も効用が高いリスタートスケジュールを計算するための

データと，スケジュールを適用する問題の性質が異なる場合，精度が高すぎることは過学習に陥ってしまう危険性とトレードオフの関係にある．そのため本研究では，全列挙法ほどの精度を持たないスケジュール候補の生成法も検討し，結果について比較する．本研究で用いるスケジュール候補の具体的な生成については，アルゴリズムの概要とは関係がなく，様々なものが考えられるため，5.1 節で詳しく説明する．

3.4.4 リスタートスケジュールの効用の推定

最後に，学習データベースから効用を計算する手順の擬似コードを，アルゴリズム 3 に示す．

Algorithm 3 Estimate_utility(S, $Numsamps$, $trainingDB$)

```
utility = 1.0
for each t_j ∈ S do
  count ← 0
  for i = 1 to NumSamps do
    data ← look up trainingDB at random
    if data.time < t_j then
      count = count + 1
    end if
  end for
  prob = count / NumSamps
  utility = utility * (1 − prob)
end for
Return utility
```

本研究では，サンプリング法を用いて効用を推定する．すなわち，サンプリング数 ($NumSamps$，例えば 100 回) を定めておき，学習データベースから $NumSamps$ 回サンプリングを行い，t 秒以内に解き終わる確率を近似的に計算する．例えば，以下のような PDB があり，$NumSapms = 3$ とした場合，3 つのデータがランダムに選ばれる．

確率推定のための SAT の PDB の具体例

$$data(A_1,\ x_1,\ seed_0) = \{\ 30\ \}$$

$$data(A_1,\ x_1,\ seed_1) = \{\ 100\ \}$$

$$data(A_1,\ x_1,\ seed_2) = \{\ 50\ \}$$

$$data(A_1,\ x_1,\ seed_3) = \{\ 10\ \}$$

$$data(A_1,\ x_2,\ seed_4) = \{\ 120\ \}$$

$$data(A_1,\ x_2,\ seed_5) = \{\ 150\ \}$$

乱択の結果，上から 3 つのデータが選ばれた場合，未知の問題インスタンス x_3 に対する時刻 $t = 100$ での効用は，$U(A,\ x_3,\ 100) = 2/3 \approx 0.67$ と推定される．

サンプリング法の特徴は，

1. 推定にかかる時間が少ない．
2. 計算時間が学習データベースの大きさに依存しない．

ことが挙げられる．1 点目は，静的リスタート戦略を用いることのメリットを活かしているといえる．すなわち，静的リスタート戦略とサンプリング法を組み合わせることで，各問題ごとに新しいスケジュールを推定するための時間を短縮している．これは，Streeter ら [26] の枠組みにはないメリットだと考えられる．2 点目は，計算時間については利点とも考えられるが，推定の精度に関してはデメリットになることも考えられる．本研究では，Nearest Neighbor 法を導入することでこの関係について部分的に検証する．Nearest Neighbor 法については，5.4 節にて述べる．サンプリング数に依るアルゴリズムの挙動についての更なる解析は，今後の課題である．

4　時間的資源制約条件におけるリスタート戦略の有効性

本章では，3 章で述べた時間的資源制約条件下での問題設定における，リスタート戦略の有効性について検討する．充足可能性問題 (SAT) を，リスタートを用いずに解いた場合とリスタートを用いて解いた場合を比較し，効用がどの程度上がるかを検証する．まずは用いる問題群とアルゴリズムについて説明し，実験設定を述べる．そして実験結果を示し，リスタートの有効性について，各問題の持つ実行時間の確率分布に注目して議論する．

4.1　実験設定

本節では，実験に用いる PDB の作成方法について述べる．ここで作成した PDB は，5, 6 章の実験でも用いる．

4.1.1　対象とする問題

SAT は NP 完全問題であるため，複数の異なる SAT ソルバの性能比較は，実験的に行うことが主流である．SAT は学問的にも応用的にも興味深い問題ドメインであるため，様々や問題インスタンスや様々な SAT ソルバが集められ，それらの性能を競う SAT Competition が開催されている [15]．SAT Competition は 19XX 年から奇数年に隔年で開催されている．2008 年，2010 年には後述する Application 部門の問題のみを対象とした SAT Race が，2012 年，2014 年には SAT Competition と同じ部門を扱う SAT Challenge が開催された．本研究で扱う SAT Competition 2013 のベンチマーク問題には，Application 部門，(Hard) Combinatorial 部門，Random 部門の 3 種類の問題カテゴリが用意され，委員らにより計 600 問の問題が選ばれた．それぞれの部門は，以下のように分けられる．

- Application 部門 : ハードウェア · ソフトウェア検証，バイオインフォマティックス，プランニング，スケジューリングなど，実用的な問題ドメインを集めた部門．
- (Hard) Combinatorial 部門 : 組み合わせ最適化問題などの中から，既存の SAT ソルバが解くのが難しい問題を集めた部門．
- Random 部門 : k-SAT において，ランダムに生成した問題を集めた部門．

本研究では，application 部門 150 問と combinatorial 部門 150 問の計 300 問を使用する．random 部門は，後述する時間制限の 300 秒以内に解けた問題がほとんどなかったため，今回の解析の対象には含めない．

4.1.2 用いる SAT ソルバ : MiniSat

現在広く使用されている SAT ソルバの 1 つに，MiniSat[7] がある．MiniSat は，既存のアルゴリズムを簡潔に記述し，拡張可能な分かりやすいソルバを提供する目的で開発された．SAT Competition では，あるアイデアが性能の改善に寄与する程度を計ることを目的として，MiniSat を改良したソルバのみを対象とした MiniSat hack トラックが用意されており，本研究の目的であるリスタート戦略の有効性の検証に適したソルバであるといえる．

MiniSat は DPLL アルゴリズム [6] を基本とする，完全な系統的アルゴリズムの一種である．高速化のために，新たな節の学習を行う CDCL(conflict-driven clause learning) 法の効率化 [30] と学習節の簡単化 [25]，単位伝搬の高速化のための監視リテラルの追加 [18]，VSIDS(variable state independent decaying sum) と呼ばれる変数選択ヒューリスティック [18]，Luby のリスタートスケジュール [17] などが実装されている．

DPLL アルゴリズムを用いるソルバは，決定変数の選択時に乱数が用いられることで実行時間が大きくばらつく．VSIDS では学習節に登場する変数に優先的に真偽割当てを行うことで性能の向上を達成したが，ある割合でランダムに変数を選択することで探索の広域性を確保することができる．本研究では，この割合に ManySAT[12] でも用いられた 0.2 という値を用いる．この結果，乱数シードの値に応じて，実行時間にばらつきが生じる．

本研究で用いるバージョンの MiniSat2.0.0 は，初期設定で Luby の普遍的リスタートスケジュール [17] が実装されており，リスタートが実行されても学習節は再利用される．そのため，各リスタートは独立していない．内部で学習しながらリスタートを行う MiniSat に対し，外部からメタ的にリスタートをして，独立した実行を行うことはできるが，本研究では混乱を避けるため，学習を行うリスタートは行わない．

4.1.3 PDB の作成

問題群の 600 問に対し，300 秒 (5 分) の時間制約の下で，MiniSat2.2.0 を 100 回実行した．100 回の実行のうち，300 秒以内に 1 度でも判定が終了した問題を，PDB の要素として用いることとする．300 秒で判定が終了しなかった場合は，実行時間 $t_{solved} = 300$ とする．Application 部門では 150 問中 97 問，Combinatorial 部門では 150 問中 92 問，計 189 問がこの条件に当てはまった．判定が終了した問題の詳細については，付録 A にまとめる．

本研究で用いる 300 秒の時間制限は，実際の SAT Competition 2013 で用いられた時間制限の 5000 秒と比較すると短く，実用的なデータとはいい難い．これは，PDB の作成の

ために，各問題インスタンスを異なる乱数シードを用いて 100 回解く (計 30000 秒制限に相当)，という条件と時間的制約のために，1 回の制限時間を短くしたことに依る．1 回の時間制限を 30 分程度とした，より実用的なデータを用いた提案手法の解析は，今後の課題である．

以上より，本研究で用いる PDB は，アルゴリズム A は MiniSat2.2.0 の，内部リスタートなし，変数決定乱数 = 0.2 の設定，問題数は 189 問，各問題に対し乱数シードの数は 100 通りの，計 18900 個のデータの組 $(A,\ x,\ seed,\ t_{solved})$ からなるデータベースとする．

4.1.4 評価の方法

本章の目的は，リスタートを用いた場合に，リスタートを用いない場合と比べてどの程度効用が上昇するかを調べることである．最適なリスタートスケジュールを推定するためには実行時間の確率分布を完全に知っている必要があるが，100 回の実行データを用いて近似的に求めることとする．最適なリスタートスケジュールは，3.4.2 節で述べたアルゴリズムにより求める．スケジュール候補の生成には，全列挙法を用いる．全列挙法のパラメータには，$T = 300$, $k = 10$ を用いた．ただし，効用の推定にはサンプリング法は用いず，100 回の効用の平均を用いる．これは，100 回の実行結果から真の確率を近似することに対応する．

リスタートを用いない場合，すなわち $S_0 = \{300\}$ の場合の効用 $U(A,\ x,\ T,\ S_0)$ を norestart utility，求めた最適リスタートスケジュール S^* を適用した場合の効用 $U(A,\ x,\ T,\ S^*)$ を restart utility と表記し，

$$R = \frac{\text{restart utility}}{\text{norestart utility}} = \frac{U(A,\ x,\ T,\ S^*)}{U(A,\ x,\ T,\ S_0)}$$

にて定義される R を用いて，効用の上昇の程度を測る．

本研究では，静的リスタート戦略の有効性の検証を目的としているが，上記のリスタートの有効性の指標は，他の適応的リスタート戦略 [1][24] などにも適用可能である．本研究の静的リスタート戦略と適応的リスタート戦略との比較は，今後の課題である．

4.2 実験結果

PDB 内に属する 189 問に，リスタートスケジュールを適用した結果を，表 4.2 に示す．表中の efficient probs は，$R \geq 1.2$ の問題数，すなわち，リスタート戦略が有効だろうと考えられる問題数を示す．ここで，R の閾値の設定に関しては任意性があり，適切な閾値の設定方法は今後の課題である．

all probs	efficient probs	ave R	max R
189	72	2.10	13.01

表 1　各問題にリスタートスケジュールを適用した結果

全問題の平均を見ると R は約 2.1 となっており，リスタートを適用すると，時間内に判定が終了する確率が 2 倍程度上昇することが分かる．また，最大では約 13 倍効用が上昇することが分かる．ここで，アルゴリズム 2 において全列挙法で生成されるスケジュール候補 $\mathfrak{S}$ には $S_0 = \{300\}$ も含まれるため，必ず $R \geq 1.0$ になることに注意したい．

一方，$R \geq 1.2$ となる問題数は 72 問であり，全体の約 38% にとどまる．この結果については，次節で考察する．

また，個々のドメインについて R を計算した結果を付録 A にまとめる．リスタートが有効なドメインもある一方，ドメイン内でリスタートの有効性がばらつくドメインもあり，一定の傾向は観測されなかった．

4.3　リスタートの有効性と効用関数の凸性

実験の結果，リスタートが効用の上昇に有効である場合，すなわち $R \geq 1.2$ となる場合は全問題の約 38% にとどまった．本節では，この原因について 2 つの観点から考察する．

4.3.1　パターン 1: $q(T) = 1.0$

1 つ目は，$U(A, x, \{300\}) = q(300) \approx 1.0$ となる場合である．すなわち，リスタート戦略を用いずとも，時間内に 100% 解き終わる場合である．効用は，時間制約である T 秒以内に解き終わる確率で定義したため，効用の最大値は 1.0 である．そのため，リスタートを用いずとも効用が 1.0 の場合，リスタート戦略は有効ではない結果となる．

典型的なドメインを，表 4.3.1 に示す．

prob-name	norestart utility	restart utility	ave schedule	R
*prime*2209 – 84	1.00	1.00	300	1.00
*prime*2209 – 85	1.00	1.00	300	1.00
*prime*2209 – 92	1.00	1.00	300	1.00
*prime*2209 – 96	1.00	1.00	300	1.00

表 2　`combinatrial/schal12-selected/satrace-unselected/repeat/prime2209` ドメイン

表中の ave schedule は，最適リスタートスケジュールのリスタート区間の長さの平均を示している．例えば，$S = \{100, 200, 300\}$ の場合，ave schedule = 200 となる．この場合，リスタートしない場合の効用が 1.0 であり，最適リスタートスケジュールも $S_0 = \{300\}$ であることが分かる．

この原因により，リスタートが有効でなかった問題は，全部で 47 問存在した．これは全体の約 25% にあたる．

4.3.2 パターン 2: 効用関数の凸性

前節の原因は自明であったが，norestart utility が 1.0 より小さくても，$R < 1.2$ となる問題は存在する．

典型的なドメインを表 4.3.2 に示す．

prob-name	norestart utility	restart utility	ave schedule	R
*md*5_47_1	0.01	0.01	300	1.0
*md*5_47_2	0.54	0.54	300	1.0
*md*5_47_3	0.5	0.54	150	1.1
*md*5_48_2	0.24	0.24	300	1.0
*md*5_48_4	0.01	0.01	300	1.0

表 3　application/satchal12-selected/SChal12/SAT_Race_unselected/mironov-zhang ドメイン

この表を見ると，norestart utility は 1.0 より小さいにも関わらず，$R < 1.2$ となる問題が存在することが分かる．また，計算された最適スケジュールも，ほとんど $S_0 = \{300\}$ となっていることが分かる．

この原因について，具体的な効用関数を用いて考える．例えば，$S_1 = \{t_1, t_2\}$ という時刻 t_1 でリスタートを 1 回行うリスタートスケジュールを考える．$\sum_{i=1}^{n} t_i \leq T$ の制約より，$t_2 = T - t_1$ とする．このとき，効用関数 $U(A, x, T, S_1)$ は，

$$\begin{aligned} U(A, x, T, S_1) = &\quad 1 - (1 - q(t_1))(1 - q(t_2)) \\ =&1 - (1 - q(t_1))(1 - q(T - t_1)) \end{aligned}$$

と t_1 の 1 変数関数でかける．ただし，$q(t)$ はアルゴリズム A が問題 x を解く実行時間の累積確率分布である．ここで，関数 $V(t)$ を

$$V(t) = (1 - q(t))(1 - q(T - t))$$

と定義する．3.3 節の考察より S_1 には対称性があるため，$0 \leq t \leq T/2$ の範囲で考えてよい．

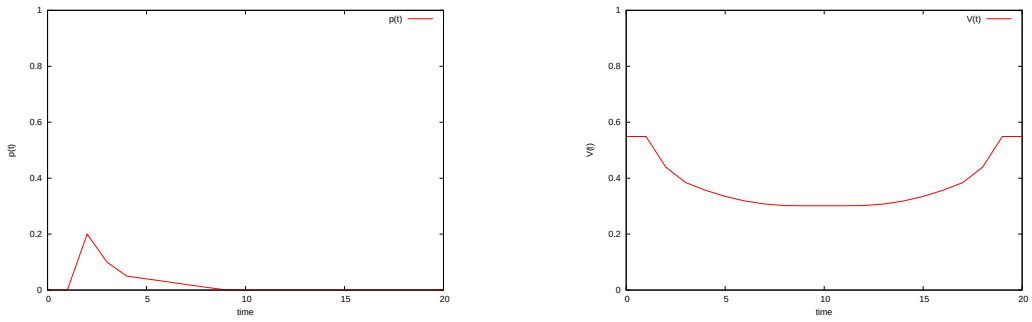
$V(t)$ が下に凸な関数の場合，$U(A, x, T, S_1)$ は t_1 に関して上に凸な関数となり，$V(t)$ が最小値をとる場合に最大値をとる．$V(t)$ は $t = T/2$ に関して対称な関数であり，かつ下に凸であるため $t = T/2$ で最小値をとる．このとき，$S^* = \{T/2, T/2\}$ が最適リスタートスケジュールであり，等分リスタートスケジュールとなる．

逆に $V(t)$ が上に凸な関数の場合，$U(A, x, T, S_1)$ は下に凸な関数となり，$V(t)$ が最小値をとる場合に最大値をとる．同様の議論により，最適リスタートスケジュールは $S^* = \{T\}$ となり，これはリスタート戦略が有効でないことを表す．

また，$V(t)$ が凸関数でない場合は，より複雑なリスタート戦略が最適なリスタートスケジュールとなると考えられる．

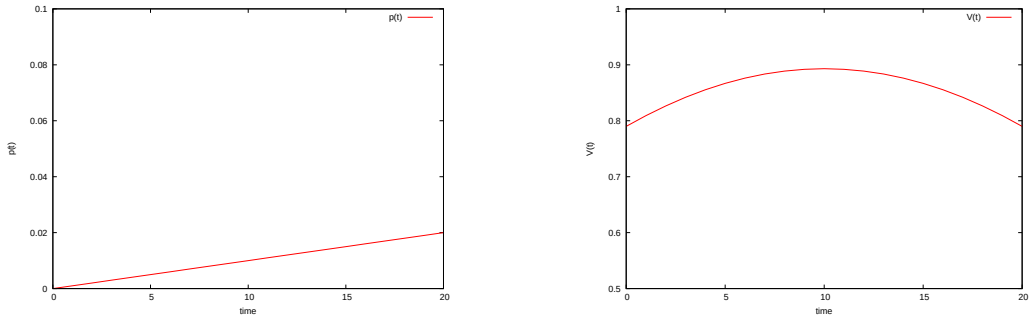
一般に，$S_{n-1} = \{t_1, t_2, \ldots, t_n\}$ の場合，上記の関数 V は $(n-1)$ 変数関数となる．$U(A, x, T, S_{n-1})$ の対称性より，各変数 t_i $(1 \leq i \leq n-1)$ の定義域は，$0 \leq t_i \leq T/n$ と考えてよい．$V(\mathbf{t})$ が下に凸な関数の場合，同様に $S = \{T/n, T/n, \ldots, T/n\}$ の等分リスタートスケジュールが最適リスタートスケジュールとなる．逆に V が下に凸な関数でない場合は，等分リスタートスケジュールが最適なリスタートスケジュールとはならない．

等分リスタートスケジュールが最適な場合と，リスタート戦略が有効でない場合の具体的な $p(t)$ と $V(t)$ の形を，それぞれ図 4.3.2 と図 4.3.2 に示す．なお，図 4.3.2 と図 4.3.2 の $p(t)$ は，$V(t)$ が凸になる条件を元に，著者が設定した．



(a) $p(t)$　　(b) $V(t)$

図 4.1　等分リスタートが最適な問題



(a) $p(t)$ (b) $V(t)$

図 4.2　リスタート戦略が有効でない問題

これは，Luby ら [17] の結果とは異なる．Luby らは，無限の時間を仮定して解析を行い，実行時間の確率分布を完全に知っている場合には，等分リスタートスケジュールが実行時間の期待値を最小化するリスタートスケジュールであることを示した．一方，時間的資源制約条件の下では，等分リスタートスケジュールは必ずしも最適なリスタートスケジュールではない．これは，有限の時間を仮定していることに起因すると考えられる．

以上の考察では，効用関数中に現れる関数 V について，上に凸，下に凸，凸でない場合の三種類に，大まかに分けて考えた．また V の引数であるベクトル $\boldsymbol{t}$ のサイズにより，V の凸性も変化することが考えられる．リスタートの有効性と効用関数についての理論的な検証は，今後の課題である．

最後に，リスタート戦略の有効性の結果を，表 4.3.2 にまとめる．表中の pattern 1 は，リスタートしない場合にすでに 100% 解き終わっていた問題数を示す．pattern 2 は，それ以外の，実行時間の確率分布から計算される効用関数の形により，リスタート戦略が有効でない問題数を示す．

all probs	efficient probs ($R \geq 1.2$)	not efficient (pattern 1)	not efficient (pattern 2)
189	72 (38%)	47 (25%)	70 (37%)

表 4　リスタート戦略の有効性のまとめ

5　PDB を用いたリスタートスケジュール推定法

本章では，PDB を用いたリスタートスケジュール推定法を充足可能性問題 (SAT) に適用し，その性能を評価する．その際に，3.4.1 節で議論したように，学習データベースをどのように選ぶか，という問題がある．また，過学習に陥らないために，学習に用いるスケジュール候補をどのように生成するか，という問題がある．前者に関して，まずは，同じドメイン内の全問題を学習データベースとして用いる場合を検討する．その後，学習データベースの範囲を広げ，Nearest Neighbor 法を導入する場合を検討する．後者に関しては，3.4.3 節で述べた全列挙法の他に定型スケジュール列挙法を導入し，両者を用いた結果を比較する．

5.1　スケジュール候補の生成:定型スケジュール列挙法

全列挙法では，k 秒の精度でリスタート区間を区切り最適なスケジュールを計算するため，学習データベース内のデータに特化したスケジュールを生成し，過学習に陥るおそれがある．本章では，学習データベースを用いて生成したリスタートスケジュールを新たな問題へ適用するため，ある程度ロバストなスケジュールを生成することが望ましい．

そのため，ManySAT[12] で用いられているリスタートスケジュールを参考に，

- 等分リスタートスケジュール
- 線形リスタートスケジュール
- 幾何リスタートスケジュール
- Luby リスタートスケジュール

の 4 つのリスタートスケジュールのみをスケジュール候補として生成する枠組みを導入する．

等分リスタートスケジュールは，常に同じ長さの区間でリスタートを繰り返すスケジュールのことである．例えば，$S = \{30, 30, 30\,30, 30, 30, 30, 30, 30, 30\}$ は等分リスタートスケジュールである．

線形リスタートスケジュールは，ある定数に比例して，リスタート区間が増えていくようなスケジュールのことである．例えば，$S = \{10, 20, 30, 40, 50, 60, 70\}$ は線形リスタートスケジュールである．

幾何リスタートスケジュールは，ある定数の 2 乗，3 乗，... に比例して，リスタート区

間が増えていくようなスケジュールのことである．例えば，S = {2, 4, 8, 16, 32, 64, 128} は幾何リスタートスケジュールである．

Luby リスタートスケジュール [17] は，どんな実行時間の確率分布を持つ問題に適用しても，最適な実行時間の $O(\log)$ 倍で実行時間が収まるような普遍的なリスタートスケジュールのことである．これは，S_l = {1, 1, 2, 1, 1, 2, 4, 1, 1, 2, 1, 1, 2, 4, 8} を基本とし，S_l の各区間を定数倍したスケジュールを指す．

以上 4 種類のスケジュールのみから，アルゴリズム 2 で用いるスケジュール候補 $\mathcal{S}$ を生成する方法を，定型スケジュール列挙法と呼ぶ．以下に，本研究で用いる定型スケジュール列挙法が生成する具体的なリスタートスケジュールをまとめる．

本研究で用いる定型スケジュール列挙法が生成するリスタートスケジュール

- 等分リスタートスケジュール

  {300}, {150, 150}, {100, 100, 100}, {50, 50, 50, 50, 50, 50}

  {30, 30, . . . , 30}, {20, 20, . . . , 20}, {10, 10, . . . , 10}

- 線形リスタートスケジュール

  {10, 20, 30, 40, 50, 60, 70}, {20, 40, 60, 80, 100}, {30, 60, 90, 120}

- 幾何リスタートスケジュール

  {2, 4, 8, 16, 32, 64, 128}

- Lubu リスタートスケジュール

  {5, 5, 10, 5, 5, 10, 20, . . . , 40, 5, 5, 10, 5, 5, 10, 20, 5, 5, 10, 5, 5, 10, 20}

  {10, 10, 20, 10, 10, 20, 40, 10, 10, 20, 10, 10, 20, 40}

  {20, 20, 40, 20, 20, 40, 80, 20, 20}

5.2 ドメイン内の全問題を PDB に選んだ場合

一般に学習を行う場合，似ている問題を選んで学習する必要がある．SAT の場合，同じ問題ドメイン内の別の問題インスタンスから学習するのがよいだろうと考えられる．また，この考えは Fukunaga[8] や Streeter ら [26] でも用いられているものである．実験は，

1. 解く対象となる問題を選ぶ
2. その問題の属するドメインの別インスタンスの結果を PDB から学習 DB として選ぶ
3. 学習 DB から最適リスタートスケジュール S^* を計算する
4. S^* を対象の問題に適用し，効用を計算する

の手順で行う．4.1.4 節と同様，リスタートしない場合の効用を norestart utility，S^* を適用した場合の効用を restart utility と表記し，

$$R = \frac{\text{restart utility}}{\text{norestart utility}}$$

で定義される R を用いて効用の上昇を測る．また，実験は，4 章においてリスタートが有効だった問題に対して行う．そもそもリスタートが有効でない場合には，リスタートスケジュールを推測して適用しても，効用は上昇しないためである．

実験の結果を，表 5.2 に示す．

	理想 restart	all schedule	common schedule
efficient probs	39	15	12
ave R	2.76794	1.13717	0.99410
max R	13.01	5.70	3.72

表 5　ドメイン内の全問題を学習に用いる場合

表中の理想 restart は，推定を行わずに，実行時間の確率分布を知っていたと仮定した場合の結果を示す．all schedule，commmon schedule はそれぞれ全列挙法，定型スケジュール列挙法の結果を示す．まず，リスタートが有効であった 72 問のうち，ドメイン内に複数の問題があったものは 39 問であった．また，この設定においては，定型スケジュール列挙法より全列挙法の方が優れていることがわかる．R の平均も 1 を超えており，リスタート戦略はやや有効だと考えられる．最大では，実行時間の確率分布を全く知らない場合でも，約 5.7 倍の効用の上昇があった．

5.3　ベンチマーク問題集の階層性

前節で用いた学習データベースの選び方は，最もナイーブなものだといえる．以降では，その他にどのような学習データベースの選び方が考えられ，その場合の効果はどうなのか

について調べる．準備として，ベンチマーク問題集の階層性について，イメージ図を図 5.1 に示す．

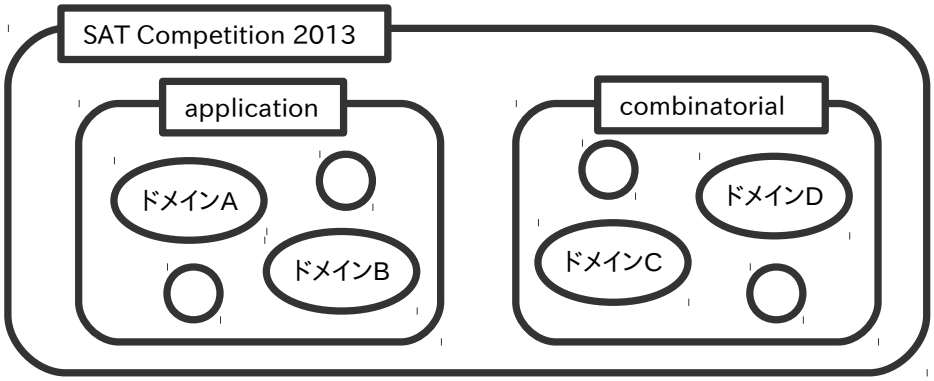


図 5.1　ベンチマーク問題集の階層性

一番大きい集合として，各問題は，SAT Competition 2013 のベンチマーク問題集に含まれている．次に大きい集合は，application 部門か combinatorial 部門である．そして，各ドメインが想定できる最も小さな集合である．

前節では，各ドメインを学習データベースとして用いた．以降では，各部門を学習データベースとして用いる場合，全問題を学習データベースとして用いる場合を検討する．前者は Xu ら [29] の枠組み，後者は Nikolic ら [20] の枠組みに対応する．

5.4　Nearest Neighbor 法

学習データベースとして，各ドメインから各部門，全問題へと拡げていくことを考えるが，学習のためのデータベースが大きくなりすぎて，学習の精度が粗くなってしまうことが考えられる．特に全問題を用いる，という考え方は，似ている問題から学習した方が結果が良くなる，という考え方とは正反対である．

この問題に対し，Nikolic ら [20] は，全問題の中から何らかの指標で似ている問題を選び，それらを用いて学習する Nearest Neighbor 法を採用した．類似度を示す指標において，上位 k 個の問題を選んで学習する方法を，k-Nearest Neighbor 法という．

本研究でも，この枠組みを用いて学習データベースを作成する．

5.5 SAT における特徴ベクトル

類似度の指標を計算する場合，問題ファイルから計算される，何らかの特徴量が必要である．Nudelman ら [21] は，SAT の問題ファイルから 91 個の特徴量を抽出し，実行時間の期待値を推定するのに重要な特徴量について調べた．Nikolic ら [20] は，91 個の中から計算の負荷が少ないものを選び，類似度の計算を行った．本研究では，Nikolic らの枠組みを用いる．その特徴量を，図 5.2 にまとめる．

問題サイズに関する特徴量:

1-3 . 節の数 c，変数の数 v，比 v / c

Variable-Clause グラフに関する特徴量:

4-8 . 変数ノードの統計量 : 平均, 変動係数, 最小値, 最大値, エントロピー

9-13 . 節ノードの統計量 : 平均, 変動係数, 最小値, 最大値, エントロピー

バランスに関する特徴量:

14-16 . 各節内の変数の + / - 比 : 平均, 変動係数, エントロピー

17-21 . 各変数の + / - 比 : 平均, 変動係数, 最大値, 最小値, エントロピー

22-23 . 2変数節, 3変数節の割合

ホーン式との類似度:

24 . ホーン節の割合

25-29 . 各変数のホーン節への出現数 : 平均, 変動係数, 最小値, 最大値, エントロピー

図 5.2 SAT の問題ファイルから計算される特徴量

図 5.2 の特徴量を並べたベクトル $\boldsymbol{x}$ を特徴ベクトルという．特徴ベクトル $\boldsymbol{x}$ は問題毎に計算され，問題間の類似度は特徴ベクトルを元に計算される．

問題インスタンス 1 と問題インスタンス 2 の最も単純な類似度は，特徴ベクトル $\boldsymbol{x_1}$ と $\boldsymbol{x_2}$ のユークリッド距離である．Nikolic ら [19] は，実行時間の期待値を推定する問題設定において，以下の式で距離を定義した方がより正確に推測できることを実証した．

$$d(\boldsymbol{x_1},\ \boldsymbol{x_2}) = \sum_i \frac{|x_i - y_i|}{\sqrt{x_i y_i} + 1}$$

本研究でも，この $d(\boldsymbol{x_1}, \boldsymbol{x_2})$ を用いて，類似度を測ることとする．

5.6 実験設定

以上をまとめると，スケジュール推定に用いる学習データベースの選び方として，

- ドメイン内の問題に k-Nearest Neighbor 法を適用したもの
- 部門内の問題に k-Nearest Neighbor 法を適用したもの
- 全問題に k-Nearest Neighbor 法を適用したもの

の 3 種類を考える．しかし，これらは人為的に部門分けしたものや，人為的に抽出した特徴量が知識として類似度に影響している．

リスタート戦略が有効かどうかは，実行時間の確率分布の形に依存することを 4.3 節で考察した．そこで本研究では，学習データベースの選び方として，もう 1 つ

- リスタート戦略が有効な問題

という学習データベースの作成方法を提案する．次節で，上記の 4 種類の学習データベースを用いて推定を行った結果をまとめる．

5.7 実験結果

5.7.1 ドメイン内で Nearest Neighbor 法を用いた場合

まず，ドメイン内で Nearest Neighbor 法を用いた場合の結果を，表 6, 5.7.1 に示す．

	理想 restart	1NN	2NN	3NN
efficient probs	26	10	7	7
ave R	2.70769	1.17884	1.15500	1.02153
max R	13.01	5.70	7.01	3.41

表 6　ドメイン内で Nearest Neighbor 法を用いる場合 (全列挙法で学習)

	理想 restart	1NN	2NN	3NN
efficient probs	26	10	7	7
ave R	2.70769	1.08038	1.00807	0.91807
max R	13.01	3.41	4.34	2.67

表 7　ドメイン内で Nearest Neighbor を用いる場合 (定型スケジュール列挙法で学習)

表 6 は全列挙法でスケジュール候補を生成し，最適スケジュールを求めた場合の結果であり，表 5.7.1 は定型スケジュール列挙法でスケジュール候補を生成し，最適スケジュールを求めた場合の結果である．kNN の表記は，k-Nearest Neighbor 法を用いたことを表す．$k = 3$ まで Nearest Neighbor 法を試すため，ドメイン内に 4 問以上解けた問題があるドメイン (計 6 ドメイン) に対して実験を行った．

結果を見ると，最もよい設定は全列挙法 + 1-Nearest Neighbor 法を用いた場合である．これは，ドメイン内で Nearest Neighbor 法を用いない場合と比べて少しよい結果といえる．また，Nearest Neighbor の数については，小さくなるほど R の平均値が大きくなる，という傾向がある．

5.7.2　部門内から Nearest Neighbor 法を用いた場合

次に，同じ部門内から Nearest Neighbor 法を用いて学習データベースを作成した場合の結果を表 8, 9 に示す．

	opt-par	1NN	2NN	3NN	4NN	5NN
efficient probs	39	13	10	12	11	11
ave R	2.76794	1.09512	0.97871	1.11128	1.18179	0.96820
max R	13.01	4.34	5.69	5.23	9.10	3.41

表 8　部門内で Nearest Neighbor を用いる場合 (全列挙法で学習)

	opt-par	1NN	2NN	3NN	4NN	5NN
efficient probs	39	12	9	9	10	9
ave R	2.76794	1.09871	1.05897	1.09307	0.97923	0.77794
max R	13.01	8.71	8.71	13.01	3.72	3.72

表 9　部門内で Nearest Neighbor を用いる場合 (定型スケジュール列挙法で学習)

まず，R の最大値について考察する．application 部門での定型スケジュール列挙法 + 3-Nearest Neighbor 法や，combinatorial 部門での定型スケジュール列挙法 + 1-Nearest Neibor 法などをみると，実行時間の確率分布を完全に知っていた場合と同じ効用の上昇が得られている問題があることがわかる．この場合，全列挙法ではなく定型スケジュール列挙法を用いたことで，過学習に陥らずに済んでいると考えられる．

また，R の平均値についても，ドメイン内の問題のみから学習データベースを作成した場合に比べて，より大きい値になっている．しかし，efficient probs の値を見ると，リスタート戦略が特に有効だった問題の R 値に，平均値が引きづられている様子がわかる．

Nearest Neighbor の数については，一定の傾向はないようにみえる．これは，ドメイン内で Nearest Neighbor 法を用いる場合とは異なる．

5.7.3　全問題の中から Nearest Neighbor 法を用いた場合

次に，全問題から Nearest Neighbor 法を用いて学習データベースを作成した場合の結果を，表 10, 11 に示す．

	理想 restart	1NN	2NN	3NN	4NN	5NN
efficient probs	39	14	10	12	12	11
ave R	2.76794	1.14615	0.95307	1.08564	1.22923	0.92820
max R	13.01	4.34	5.69	5.23	9.10	3.41

表 10　全問題内で Nearest Neighbor を用いる場合 (全列挙法で学習)

	理想 restart	1NN	2NN	3NN	4NN	5NN
efficient probs	39	12	9	8	10	8
ave R	2.76794	1.12435	1.03333	1.06769	0.95358	0.76000
max R	13.01	8.71	8.71	13.01	3.72	3.72

表 11　全問題内で Nearest Neighbor を用いる場合 (定型スケジュール列挙法で学習)

まず，R の最大値に着目すると，定型スケジュール列挙法 + 3-Nearest Neighbor 法の場合に，最適リスタートスケジュールと同じ効用の上昇が得られていることがわかる．しかし，efficient probs の数をみると，全列挙法の方が多くの問題で $R \geq 1.2$ となっていることがわかる．また，R の平均値についても，全列挙法を用いたほうがやや高くなっている．

特に，全列挙法 + 4-Nearest Neighbor 法をみると，$R \geq 1.2$ となっており，平均的に，推測されたリスタートスケジュールが有効になっている．

5.8　リスタート戦略が有効な問題を学習データベースに選んだ場合

最後に，リスタート戦略が有効な問題群から学習データベースを作成した場合の結果を示す．リスタート戦略の有効性を $R \geq 1.2$ で定義したものを表 12, $R \geq 2.0$ で定義したものを表 13 に示す．

	理想 restart	all schedule	common schedule
efficient probs	39	16	13
ave R	2.76777	1.25873	1.16461
max R	13.01	5.25	4.78

表 12　$R \geq 1.2$ の問題を学習に用いる場合

	理想 restart	all schedule	common schedule
efficient probs	14	9	7
ave R	4.82053	2.58540	2.95361
max R	13.01	10.72	13.01

表 13　$R \geq 2.0$ の問題を学習に用いる場合

表 12 をみると，全列挙法を用いて学習した場合に，R の平均値が 1.2 を超えており，平均的にリスタート戦略が有効だといえる．

表 13 では，R の平均値が 2 を超えており，平均的に 2 倍以上の効用の上昇がみられる．また，実行時間の確率分布を事前に知っていた場合の R の平均値は約 5 倍となっており，リスタート戦略が 2 倍以上の効用の上昇をもたらすような問題の場合には，それらの問題から学習したリスタートスケジュールを適用しても大きな効用の上昇を得られることがわかる．

以上の結果より，問題を解く前にその問題がリスタート戦略が有効な問題だと判定できれば，リスタート戦略が有効であった問題群からリスタートスケジュールを学習し，適用することが有効である，という仮説が考えられる．次章では，問題の持つ特徴から，リスタート戦略の有効性が事前に判定可能かどうかを考える．

6　リスタート戦略の有効性の推定

本章では，リスタート戦略の有効性をを，事前に判定可能かどうかについて考える．本研究ではこの問題に対し，リスタート戦略が有効であることを *good*，リスタート戦略が有効でないことを *bad* とラベル付けし，問題ファイルから計算される特徴ベクトルを用いたクラス分類問題として定式化する．まず，クラス分類問題について述べ，それを解くサポートベクターマシンについて説明する[*4]．そして，検証方法を説明し，最後に結果をみる．

6.1　クラス分類問題とサポートベクターマシン

クラス分類問題とは，一般にある入力ベクトル $\boldsymbol{x}$ が渡されたとき，ある出力 y を返す問題のことである．入力ベクトル $\boldsymbol{x}$ は，特徴ベクトルとも呼ばれる．本研究では，出力 y は，リスタート戦略が有効である *good* か，有効でない *bad* かだけなので，最も単純な 2 値クラス分類に属する．

2 値クラス分類では，特徴ベクトル $\boldsymbol{x}$ と実数値関数 $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$ を用いて，$y = f(\boldsymbol{x})$ の値により，$y \geq 0$ ならば $\boldsymbol{x}$ を正のクラス，$y < 0$ ならば $\boldsymbol{x}$ を負のクラスに分類する．例えば，

$$f(\boldsymbol{x}) = \mathrm{sign}(\boldsymbol{w} \cdot \boldsymbol{x} - h)$$

の場合，線形クラス分類と呼ばれ，ニューラルネットワークや統計学の分野で多くの研究がなされてきた．ただし，$\boldsymbol{w}$ は重みベクトルであり，h は分類の超平面を定める閾値である．$f(\boldsymbol{x})$ は線形判別器やパーセプトロン，線形識別関数などと呼ばれる．

線形クラス分類は理解しやすく，扱いやすいモデルであるが，線形分離可能なデータであることが仮定されており，表現力が弱い．一方サポートベクターマシンは，1990 年代にカーネル法と組み合わされ，非線形の識別関数が構成できるように拡張された．また，訓練データの過学習に陥らないための汎化能力に関して，マージン最大化という基準を用いて実現している．これらの工夫により，サポートベクターマシンは認識性能の優れた学習モデルの 1 つとなっている．

本研究では，サポートベクターマシンを用いて，2 値クラス分類を行う．

[*4] [5] を参考にまとめた．

6.2 k-分割交差検証

クラス分類器の精度を確かめる場合に最も単純な方法は，手元のデータセットを 2 つに分け，一方を訓練セット，もう一方をテストセットとし，訓練セットを用いて作成したクラス判別器をテストセットに当てはめ，精度を検証することである．ただしこの方法は，手元のデータ数が少ない場合には訓練セットやテストセットが少なくなってしまい，学習の精度や検証の精度が落ちてしまうという問題がある．

この問題を回避するために，k-分割交差検証という手法を用いる．まず，手元のデータを k 個に分割し，そのうちの 1 つをテストセットに選ぶ．残りの全データを訓練セットとして学習を行い，正答率を計算する．これを k 回繰り返し，k 回の正答率の平均値を全体の正答率とする．以上の操作により，訓練セットのデータ数や検証の精度を保証する．

本研究では，この k-分割交差検証を用いてリスタート戦略の有効性の正答率を計算し，正答率をリスタート戦略の有効性の推定可能性と考える．

6.3 実験設定

解析に用いている 189 問の問題中，特徴ベクトルが計算できたのは 171 問であった．この 171 問の問題に対し，*good* と *bad* のラベルをつける．

good の定義には，5.8 節と同様に，$R \geq 1.2$ の場合と，$R \geq 2.0$ の場合の 2 種類を考える．

6.4 実験結果

まず，$R \geq 1.2$ を *good* と定義した場合の結果を，表 14 に示す．

k-fold	accuracy
2	80.6%
3	81.8%
8	83.5%

表 14　$R \geq 1.2$ を *good* とした場合の正答率

表 14 中の k-fold は，交差検証の際の分割数を表す．accuracy は正答率の平均値を表す．

この結果より，80% 以上の確率で，SAT の問題ファイルから計算される特徴ベクトルから，リスタート戦略の有効性が推定できることがわかる．

次に，$R \geq 2.0$ を *good* と定義した場合の結果を，表 15 に示す．

k-fold	accuracy
2	86.5%
3	86.5%
8	86.5%

表 15　$R \geq 2.0$ を *good* とした場合の正答率

結果をみると，正答率の平均は 85% を超えており，こちらの場合の方がより正確に有効性を推定することができている．

6.5　時間的資源制約条件下でのリスタート戦略の適用アルゴリズム

前節の結果より，リスタート戦略を適用した場合の効用の上昇が 2 倍以上，すなわち $R \geq 2.0$ をリスタート戦略が有効と定義した場合には，問題を解く前にリスタート戦略の有効性が 85% 以上の精度で判定できることがわかった．

これを踏まえると，時間的資源制約条件下において，PDB を用いてリスタートスケジュールを推定し，アルゴリズム A を用いて SAT の問題インスタンス x を解く手法は以下のようになる．

Algorithm 4 Solve(A, x, PDB, $NumSamps$, T, k)

$y \leftarrow$ Estimate_restart_efficiency(x)
if y = *good* **then**
　$trainingDB \leftarrow$ Extract_trainingDB(PDB)
　$S^* \leftarrow$ Find_best_schedule($trainingDB$, $NumSamps$, T, k)
else
　$S^* \leftarrow \{T\}$
end if
Solve x using A with restart schedule S^*

$R \geq 2.0$ となる問題は，推定されたリスタートスケジュールを適用した際の R も 2.0 以上となるため，全体の効用もリスタートを全くしない場合と比べて上昇すると考えられる．

7 結論

7.1 本研究の結果

本研究では，過去の実行記録のデータベース (Past performance Data Base; PDB) を用いてリスタートスケジュールを推定する方法を，充足可能性問題 (SAT) に適用した．先行研究 [8] との相異点は，遺伝的アルゴリズムは近似アルゴリズムであるため連続的に効用が定義可能であるのに対し，SAT ソルバは連続的に効用を測るのが困難なことである．本研究では，時間的資源制約という条件の下で，ある時間 T 秒以内に問題を解く確率を最大化する，という問題設定をし，効用を連続的に定義した．その結果，解き終わる時間の期待値を最小化する場合の問題設定と比較し，スケジュール候補を生成するための探索空間は $O(n^n)$ 削減された．この利点を活かし，スケジュール候補の生成法には全列挙法を用いた．また，過学習に陥る可能性を考え，標準的なリスタートスケジュールのみをスケジュール候補として生成する定型スケジュール列挙法を導入した．実験の結果，全体的に全列挙法の方が効用の上昇度を示す R 値の平均が高く，本研究の設定においては過学習には陥らなかったことがわかる．

4 章では，各問題インスタンスの実行時間の確率分布を完全に知っているという仮定の下で，全列挙法を用いて最適なリスタートスケジュールを求め，リスタート戦略の有効性を検討した．実験の結果，対象とした問題の 189 問中，約 38% の 72 問において，リスタート戦略が有効だといえることがわかった．リスタート戦略が有効でない場合には，主に 2 つのパターンがあると考えられる．1 つは，リスタート戦略を用いない場合に，すでに 100% の確率で問題が解き終わる場合である．もう 1 つは，実行時間の確率分布の形に依る場合である．リスタートスケジュールを適用した場合の効用関数は，実行時間の確率分布を用いて記述できる．リスタートスケジュールが $S = \{t_1, t_2, \ldots, t_n\}$ の場合，効用関数 $U(A, x, S)$ は (n-1) 変数関数となるが，少なくとも U が上に凸な関数の場合は，リスタート戦略が有効でないことがわかった．

5 章では，リスタート戦略が有効であった問題について，PDB を用いて推定されたリスタートスケジュールが有効であるかどうかを検討した．その際に，学習データベースの選び方に注目し，ドメイン内から学習する方法，部門内から学習する方法，全問題から学習する方法，リスタート戦略が有効な問題から学習する方法を比較した．また，Nearest Neighbor 法を用いて，学習データベースが大きくなりすぎることに対応した．実験の結果，リスタート戦略が有効な問題から学習し，特にそれを効用の上昇 $R \geq 2.0$ となるよう

な問題に適用する場合に，有効であることがわかった．

以上の結果より，問題を解く前にリスタート戦略の有効性が判定できれば，その問題に対してのみリスタート戦略を用いる，という戦略が考えられる．6 章では，問題ファイルから計算される特徴ベクトルを用いて，リスタート戦略の有効性が判定可能かどうかについて検討した．この問題を，[*good*, *bad*] の 2 値クラス分類問題として定式化し，サポートベクターマシンを用いて解いた．実験の結果，$R \geq 1.2$ を *good* とした場合には約 80%，$R \geq 2.0$ を *good* とした場合には約 85% の正答率でリスタートの有効性が判定可能であることがわかった．

7.2 今後の課題

本研究で PDB として用いた実験データは，minisat を 300 秒の時間制限の下で解いた場合のデータである．これは，実際の SAT Competition 2013 の時間制限の 5000 秒と比較すると短く，実用的なデータとはいい難い．これは，PDB の作成のために，各問題インスタンスを異なる乱数シードを用いて 100 回解く (計 30000 秒制限に相当)，という条件と時間的制約のために，1 回の制限時間を短くしたことに依る．1 回の時間制限を 30 分程度とした，より実用的なデータを用いた提案手法の解析は，今後の課題である．

また，時間的資源制約条件の問題設定は，クラスタなどの計算環境で用いられるジョブスケジューラにジョブを投入することを想定したものである．そのため，時間制約 T を 300 秒にした場合の効用を最大化するものより，効用が 95% や，98% になる時間制約 t を最小化するものの方がより実用的であると考えられる．この方向性での研究も，今後の課題である．

また，本研究で対象とした問題インスタンスは SAT Competition 2013 のベンチマーク問題集であるが，他にも多くの問題がベンチマーク問題として存在する．さらに，本研究で用いた問題設定とアルゴリズムは，SAT に限らず，探索途中の効用を定義するのが困難なアルゴリズム全てに応用可能である．例えば，組合せ最適化問題を解くアルゴリズムやプランニング問題の，最適コストとなるプランを求めるアルゴリズム，整数計画問題の最適解を求めるアルゴリズムなどが当てはまる．本研究の戦略が他の分野でどの程度有効であるかの検証は，今後の課題である．

参考文献

[1] Armin Biere. Adaptive restart control for conflict driven sat solvers. In *Proceeding 11th Internatinal Conference on Theory and Applications of Satisfiability Testing (SAT'08)*, Vol. 4996 of Lecture Notes in Computer Science, pp. 28–33, 2008.

[2] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. Symbolic model checking without bdds. In *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, TACAS '99, pp. 193–207, 1999.

[3] Amanda Coles, Andrew Coles, Angel Garcia Olaya, Sergio Jimenez, Carlos Linares Lopez, Scott Sanner, and Sungwook Yoon. A survey of the seventh international planning competition. *AI Magazine*, Vol. 33, No. 1, pp. 83–88, 2012.

[4] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pp. 151–158, 1971.

[5] Nello Cristianini, John Shawe-Taylor, (大北剛訳). サポートベクターマシン入門. 共立出版, 2005.

[6] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Commun. ACM*, Vol. 5, No. 7, pp. 394–397, 1962.

[7] Niklas Een and Niklas Sorensson. An extensible sat-solver. In *Theory and Applications of Satisfiability Testing*, Vol. 2919 of lecture notes in computer science, pp. 502–518. Splinger, 2004.

[8] Alex S. Fukunaga. Restart scheduling for genetic algorithms. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature(PPSN-V)*, Vol. 1498 of Lecture Notes in Computer Science, pp. 357–366, 1998.

[9] Carla P. Gomes, Bart Selman, Nuno Crato, and Henry Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of Automated Reasoning*, Vol. 24, No. 1-2, pp. 67–100, 2000.

[10] Carla P. Gomes, Bart Selman, and Henry Kautz. Boosting combinatorial search through randomization. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence (AAAI98)*, pp. 431–437,

1998.

[11] Yiyuan Gong and Alex Fukunaga. Distributed island-model genetic algorithms using heterogeneous parameter settings. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pp. 820–827, 2011.

[12] Youssef Hamadi and Lakhdar Sais. Manysat: a parallel sat solver. *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)*, Vol. 6, pp. 245–262, 2009.

[13] Malte Helmert, Gabriele Roger, and Erez Karpas. Fast downward stone soup. In *Seventh International Planning Competition (IPC2011), Deterministic Part*, pp. 38–45, 2011.

[14] Bernardo A. Huberman, Rajan M. Lukose, and Tad Hogg. An economics approach to hard computational problems. *Science*, Vol. 275, pp. 51–54, 1997.

[15] Matti Järvisalo, Daniel Le Berre, Olivier Roussel, and Laurent Simon. The international SAT solver competitions. *AI Magazine*, Vol. 33, No. 1, pp. 89–92, 2012.

[16] Jon Kleinberg and Eva Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., 2005.

[17] Michael Luby, Alistair Sinclair, and David Zuckerman. Optimal speedup of las vegas algorithms. *Information Processing Letters*, Vol. 47, No. 4, pp. 173–180, 1993.

[18] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient sat solver. In *Annual ACM IEEE Design Automation Conference*, pp. 530–535, 2001.

[19] Mladen Nikolic, Filip Maric, and Predrag Janicic. Instance-based selection of policies for sat solvers. In *Proceeding 11th Internatinal Conference on Theory and Applications of Satisfiability Testing (SAT'09)*, Vol. 5584 of lecture notes in computer science, pp. 438–452, 2009.

[20] Mladen Nikolic, Filip Maric, and Predrag Janicic. Simple algorithm portfolio for sat. *Artificial Intelligence Review*, Vol. 40, No. 4, pp. 457–465, 2013.

[21] Eugene Nudelman, Kevin Leyton-Brown, Holger H. Hoos, Alex Devkar, and Yoav Shoham. Understanding random sat: Beyond the clauses-to-variables ratio. In *Tenth Internatioal Conference on Principles and Practice of Constraint Programming (CP 2004)*, Vol. 3258 of Lecture Notes in Computer Science, pp. 438–452, 2004.

[22] Fei Peng, Ke Tang, Guoliang Chen, and Xin Yao. Population-Based Algorithm Portfolios for Numerical Optimization. *IEEE Transactions on Evolutionary Computation*, Vol. 14, No. 5, pp. 782–800, 2010.

[23] Stuart Russell and Peter Norvig. エージェントアプローチ人工知能. 共立出版, 2008.

[24] Carsten Sinz and Markus Iser. Problem-sensitive restart heuristics for the dpll procedure. In *Proceeding 12th Internatinal Conference on Theory and Applications of Satisfiability Testing (SAT'09)*, Vol. 5584 of Lecture Notes in Computer Science, pp. 356–362, 2009.

[25] Niklas Sörensson and Armin Biere. Minimizing learning clauses. In *Proceeding 12th Internatinal Conference on Theory and Applications of Satisfiability Testing (SAT'09)*, Vol. 5584 of Lecture Notes in Computer Science, pp. 237–243, 2009.

[26] Matthew Streeter, Daniel Golovin, and Stephen F. Smith. Restart schedules for ensembles of problem instances. In *Proceedings of the Twenty-Second Conference on American Association for Artificial Intelligence (AAAI)*, pp. 1204–1210, 2007.

[27] Matthew Streeter and Stephen F. Smith. New techniques for algorithm portfolio design. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence(UAI)*, pp. 519–527, 2008.

[28] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp. 67–82, 1997.

[29] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Satzilla: Portfolio-based algorithm selection for sat. *Journal of Artificial Intelligence Research(JAIR)*, Vol. 32, No. 1, pp. 565–606, 2008.

[30] Lintao Zhang, Conor F. Madigan, Matthew H. Moskewicz, and Sharad Malik. Efficient conflict driven learning in a boolean satisfiability solver. In *Proceedings of the 2001 IEEE/ACM International Conference on Computer-aided Design*, ICCAD '01, pp. 279–285, 2001.

[31] 井上克巳, 田村直之. Sat ソルバーの基礎 (< 特集 > 最近の sat 技術の発展). 人工知能学会誌, Vol. 25, No. 1, pp. 57–67, 2010.

付録 A　各ドメインにおけるリスタートの有効性

A.1　application 部門

domain name(number of solved problems)	ave nopar util	ave par util	ave R
nossum--sha1/23-96(2)	0.04	0.14	3.17
nossum--sha1/22-160(6)	0.03	0.08	3.69
nossum--sha1/22-128(10)	0.12	0.25	2.11
nossum--sha1/22-144(6)	0.06	0.09	1.93
nossum--sha1/23-80(1)	0.04	0.08	1.96
nossum--sha1/23-64(2)	0.15	0.23	1.29
soos--grainofsalt/hitag2-exptime9600(1)	0.06	0.06	1.00
mayer-eichberger--carseq(2)	0.02	0.02	1.99
kummling_grosmann--pesp(2)	0.48	0.47	1.00

表 16　application 部門, SC13_submissons ドメイン

domain name(number of solved problems)	ave nopar util	ave par util	ave R
application/SAT2012_AES/Benchmarks/24(1)	0.81	0.81	1.00

表 17　application 部門, satchal12-submitted ドメイン

domain name(number of solved problems)	ave nopar util	ave par util	ave R
SAT11/kullmann/AES/Benchmarks/64(1)	0.02	0.02	1.00

表 18　application 部門, sat11-selected ドメイン

domain name(number of solved problems)	ave nopar util	ave par util	ave R
SAT_Race_2010/software-verification/post(1)	1.00	1.00	1.00
SAT_Race_2010/cryptography/mizh(3)	0.29	0.30	1.01
SAT_Race_2010/cryptography/desgen(2)	0.04	0.04	1.00
SAT_Race_2008(3)	0.42	0.52	1.87
SChal12/SAT_Race_unselected/mironov-zhang(5)	0.26	0.27	1.02
SChal12/SAT09/crypto/md5gen(1)	0.45	0.45	1.00
SChal12/SAT09/crypto/desgen(1)	0.10	0.26	2.60
SChal12/SAT11/rintanen/SATPlanning(7)	0.37	0.40	1.05
SAT07/industrial/velev/SAT4.0(6)	0.65	0.93	1.46
SAT07/industrial/babic/xinetd(2)	1.00	1.00	1.00
SAT07/industrial/anbulagan/medium-sat(3)	0.11	0.18	1.65
SAT07/industrial/anbulagan/hard-sat(1)	0.03	0.06	1.97
SAT07/industrial/fuhs/medium(1)	0.05	0.05	1.00
SAT07/industrial/grieu(2)	0.13	0.16	1.17
SAT09/application/aprove09(1)	0.03	0.04	1.32
SAT09/application/crypto/desgen(2)	0.16	0.16	1.49
SAT09/application/SAT_RACE06(1)	0.07	0.20	2.89
SC2012_Application/satrace-unselected/zeus(1)	0.14	0.14	1.00
SAT11/SAT09/APPLICATIONS/crypto/desgen(2)	0.02	0.14	7.01
SAT11/SAT09/APPLICATIONS/bioinfo(1)	0.69	0.69	1.00
SAT11/SAT07/industrial/vliw_sat_4.0(1)	0.51	0.99	1.94
SAT11/SAT_RACE06(1)	0.13	0.61	4.70
SAT11/leberre/2dimensionalstrippacking(3)	0.38	0.47	1.61
SAT11/fuhs/slp-synthesis-AES(2)	0.01	0.01	1.49
SAT11/rintanen/SATPlanning(2)	0.09	0.36	2.58
SAT11/jarvisalo/AAAI2010-SATPlanning(2)	0.56	0.57	1.14
SAT11/SAT05/industrial/grieu05/vmpc(1)	0.17	0.46	2.69

表 19　application 部門, satchal12-selected, (SAT+UNSAT) ドメイン

A.2 combinatorial 部門

domain name(number of solved problems)	ave nopar util	ave par util	ave R
mugrauer_balint--LABS(15)	0.04	0.08	2.21
kovasznai_frohlich_biere--smtbench/ndist.b(6)	0.18	0.18	1.00
surynek--gridMRPP/8x8_10pc/SAT(3)	0.35	0.35	1.00
surynek--gridMRPP/6x6_10pc/SAT(8)	0.42	0.42	1.25
surynek--gridMRPP/6x6_10pc/UNSAT(1)	0.21	0.21	1.00
surynek--gridMRPP/4x4_10pc/SAT(8)	0.99	1.00	1.01
manthey--hidoku(1)	0.01	0.01	1.00
biere--tph(1)	1.00	1.00	1.00
bebel_yuen--factoring(7)	0.69	0.73	1.10
heule--randomMUS(9)	0.78	0.78	1.00
heule_szeider--cwd/famous(2)	1.00	1.00	1.00
heule_szeider--cwd/rndcwd(1)	1.00	1.00	1.00

表 20　comblication 部門, SC13_submissions ドメイン

domain name(number of solved problems)	ave nopar util	ave par util	ave R
hard_combinatorial/EnsembleComputation(1)	0.02	0.05	2.45

表 21　comblication 部門, schal12-submitted ドメイン

domain name(number of solved problems)	ave nopar util	ave par util	ave R
SAT05/jarvisalo05/mod2c-rand3bip-sat(1)	0.01	0.01	1.00

表 22　comblication 部門, sat11-selected ドメイン

domain name(number of solved problems)	ave nopar util	ave par util	ave R
crafted/SAT09/CRAFTED/modcircuits(1)	0.05	0.26	5.23

表 23　comblication 部門, sat09-selected ドメイン

domain name(number of solved problems)	ave nopar util	ave par util	ave R
crafted/Difficult/contest05/others(1)	0.05	0.26	5.25

表 24　comblication 部門, sat07-selected ドメイン

domain name(number of solved problems)	ave nopar util	ave par util	ave R
SAT07/crafted/Difficult/contest-02-03-04(1)	0.53	0.53	1.00
SAT07/crafted/Difficult/contest05/QG(1)	1.00	1.00	1.00
SAT07/crafted/Hard/contest04/connamacher(1)	0.04	0.04	1.00
SAT07/Hard/contest05/QG(1)	0.58	0.58	1.00
SAT07/Medium/contest05/others(1)	1.00	1.00	1.00
SAT07/Medium/contest05/pebbling(1)	0.15	0.16	1.06
SAT07/Medium/contest05/jarvisalo(1)	0.91	0.91	1.00
SChal12/satrace-unselected/repeat/prime2209(4)	1.00	1.00	1.00
SChal12/SAT09/rbsat/random/unforced(2)	0.07	0.07	1.00
SChal12/combinatorial_sat/EnsembleComputation(4)	0.11	0.21	3.30
SChal12/SAT11/kullmann/VanderWaerden_pd_3k(1)	1.00	1.00	1.00
SAT09/edgematching/fbcolors(1)	0.99	1.00	1.01
SAT09/edgematching/compact(2)	0.57	0.59	1.11
SAT09/edgematching/explicit(3)	0.43	0.50	1.45
SAT09/edgematching/all(2)	1.00	1.00	1.00
SAT09/sgen/sat(1)	1.00	1.00	1.00
SAT09/rbsat/random/unforced(1)	0.03	0.03	1.00
SAT11/ramseycube(1)	0.99	1.00	1.01
SAT11/spence/sgen(1)	1.00	1.00	1.00
SAT11/kullmann/VanderWaerden_pd_3k(1)	1.00	1.00	1.00

表 25　comblication 部門, schal12-selected(SAT+UNSAT) ドメイン

付録 B　$R_{opt} \geq 2.0$ となる問題にスケジュール推定法を適用した結果

B.1　ドメイン内で Nearest Neighbor 法を用いた場合

	opt-par	1NN	2NN	3NN
efficient probs	11	4	2	2
ave R	4.28545	1.97272	1.86272	1.54090
max R	13.01	5.70	7.01	3.41

表 26　ドメイン内で Nearest Neighbor を用いた all schedule で学習 ($R \geq 2.0$)

	opt-par	1NN	2NN	3NN
efficient probs	11	4	3	2
ave R	4.28545	1.52181	1.45000	1.43909
max R	13.01	3.41	4.34	2.67

表 27　ドメイン内で Nearest Neighbor を用いた common schedule で学習 ($R \geq 2.0$)

B.2　部門内で Nearest Neighbor 法を用いた場合

	opt-par	1NN	2NN	3NN	4NN	5NN
efficient probs	14	4	3	4	4	4
ave R	4.82000	1.85071	1.64642	2.03000	2.10285	1.71571
max R	13.01	4.34	5.69	5.23	9.10	3.41

表 28　部門内で Nearest Neighbor を用いた all schedule で学習 ($R \geq 2.0$)

	opt-par	1NN	2NN	3NN	4NN	5NN
efficient probs	14	3	2	2	5	2
ave R	4.82000	2.01714	1.85214	2.16928	1.75357	1.18642
max R	13.01	8.71	8.71	13.01	3.72	3.72

表 29　部門内で Nearest Neighbor を用いた common schedule で学習 ($R \geq 2.0$)

B.3　全問題で Nearest Neighbor 法を用いた場合

	opt-par	1NN	2NN	3NN	4NN	5NN
efficient probs	14	4	3	4	4	3
ave R	4.82000	1.92142	1.57500	2.03000	2.17357	1.58142
max R	13.01	4.34	5.69	5.23	9.10	3.41

表 30　全問題で Nearest Neighbor を用いた all schedule で学習 ($R \geq 2.0$)

	opt-par	1NN	2NN	3NN	4NN	5NN
efficient probs	14	3	2	2	4	3
ave R	4.82000	1.94571	1.71000	2.16928	1.61285	1.25642
max R	13.01	8.71	8.71	13.01	3.72	3.72

表 31　全問題で Nearest Neighbor を用いた common schedule で学習 ($R \geq 2.0$)