数理解析研究所講究録
第1726巻 2011年 184-198

ある在庫管理問題に対する分枝限定法の応用

小島義弘 † (Yoshihiro Kojima)　山本有作 ‡ (Yusaku Yamamoto)
今堀慎治 † (Shinji Imahori)　張紹良 † (Shao-Liang Zhang)

† 名古屋大学　大学院工学研究科　計算理工学専攻
(Department of Computational Science and Engineering Graduate School of Engineering,Nagoya University)

‡ 神戸大学大学院 工学研究科　情報知能学専攻
(Department of Computer and System Engineering Graduate School of faculty of Engineering,Kobe University)

1 概要

本稿では，ある在庫管理問題の分枝限定法に基づく近似解法の提案・評価およびその改良について報告する．この問題は組合せ最適化問題であり，動的計画法を用いて解けるが，実用的な時間で解を得るにはグラフィックスプロセッサなど特別な演算加速用ハードウェアが必要である [1]．そこで，本研究ではこの問題に対して組合せ最適化問題に対する汎用アルゴリズムとして知られる分枝限定法の適用を考える．しかし，全期間に対して単純に分枝限定法を適用すると，最適解を得ることは可能であるが，計算時間を実用的なものにするようなアルゴリズムの設計は非常に困難であることが予想される．このため，対象とする問題の性質に基づく考察と，分枝限定法の計算量についての考察を基に，分枝限定法を全期間よりも短い部分期間に複数回適用するという近似解法を採用した．この近似解法により汎用 PC を用いて実用的な時間で解を得ることに成功したが，この解法においても問題例やパラメータによっては計算時間が過大になってしまう場合がある．今回は，この問題を解決するために近似解法において繰り返し用いる分枝限定法について，問題の性質に基づいた改良を加えることで更なる計算時間の短縮を目指す．

2 対象とする問題について

2.1 ある在庫管理問題

いま，ある原料を格納する K 個の倉庫があるとする．各倉庫からは毎日それぞれ決まった量だけ原料が搬出される．一方，原料を積載したトラックが毎日 1 台やって来て，K 個の倉庫のいずれか 1 つにその原料を搬入する．これを N 日にわたって行う．すると，各倉庫内の原料の充填率 (0%〜100%) は毎日変化するが，これは各日においてトラックが運んでくる原料をどの倉庫に搬入するかという選択の関数となる．今回扱う問題は，全ての倉庫の充填率が 0% からも 100% からもなるべく離れた中間の領域で推移するように，各日のトラックの搬入先を期間の最初の時点で計画する問題である．

なお，本問題を現実に適用する際には，1 日に複数のトラックが来る場合，日によって特定の倉庫に原料を搬入できない場合など，より複雑な状況が生じうる．しかし，それらへの対処は比較的容易であるため，以下でも上記の簡略化した形で説明を行う．

2.2 最小化問題への帰着

この問題を数学的に定式化するために，次のように定数・変数を定義する．なお，添字の範囲は特に指定がない場合には $1 \le k \le K$，$1 \le t \le N$ とする．

$f_k^{(t)}$ ：倉庫 k の $t\,(0 \le t \le N)$ 日目最終時点における充填率
v_k ：倉庫 k の容量
$c_k^{(t)}$ ：倉庫 k からの t 日目における搬出量
$a^{(t)}$ ：t 日目に来るトラックの積載量
$g_k(f_k)$ ：充填率が f_k のときの倉庫 k のペナルティ関数
$j(t)$ ：t 日目に来るトラックの行き先の倉庫番号

ここで，$f_k^{(0)}$，v_k，$c_k^{(t)}$，$a^{(t)}$ は与えられた定数である．関数 $g_k(f_k)$ は，$f_k = 0$（倉庫が空），$f_k = 1$（倉庫が満杯）で大きな値をとり，$f_k = 0.5$ 付近で小さな値を取る図 1 のような非負の関数である．

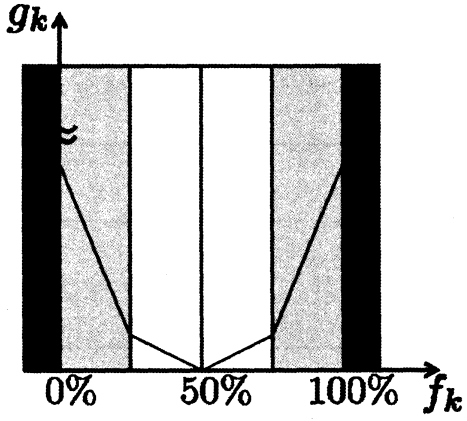

図 1: ペナルティ関数

このとき，$f_k^{(t)}(1 \le t \le N)$ は次の式により定まる．

$$f_k^{(t)} v_k = f_k^{(t-1)} v_k - c_k^{(t)} + a^{(t)} \delta_{k,j(t)} \quad (1 \le k \le K, 1 \le t \le N)$$

この式は第 t 日目の倉庫 k における原料の保存則を表す．ただし，$\delta_{k,j(t)}$ はクロネッカーのデルタである．今回扱う問題は，上記の制約式の下で目的関数

$$G\left(\{f_k^{(0)}\}_{k=1}^K; \{j(t)\}_{t=1}^N\right) \equiv \sum_{k=1}^{K} \sum_{t=1}^{N} g_k(f_k^{(t)})$$

を最小化する問題として定式化される．また，独立変数は $j(t)$ のみである．この問題については，PARTITION 問題 [2] への帰着により NP 困難であることが分かっている．

3 近似解法の提案

3.1 分枝限定法

分枝限定法はほとんど全ての組合せ最適化問題に適用できる幅広い計算原理である．今回取り扱う在庫管理計画問題も前章で制約付き組合せ最適化問題の形に定式化したので，分枝限定法を適用することがで

きる．今回扱う問題は図2の様な探索木を持ち，探索木の末端の葉の数は最大でK^N程度となることが考えられる．実用規模の問題では$K = 5$，$N = 90$程度のサイズを扱うため，これは非常に膨大な数になる．全探索では探索木のすべての葉をくまなく探索し，その中から最適解を見つけることになるが，計算量が非常に多く，実用的な時間で最適解を求めることは困難である．これに対して，分枝限定法では，基本的には全探索と同じように探索木の探索を行っていくが，その途中で限定操作と呼ばれる操作を行う．限定操作とは，探索を行っていく途中でそこから先を探索しても最適解に到達し得ないと判明した時点で，末端まで探索が終わっていなくても探索を打ち切るという処理である．(本稿ではこれ以降この限定操作のことを,「枝刈り」と呼ぶことにする.) この枝刈りにより，分枝限定法では全探索よりも探索する枝の数，つまり，計算量を減らしつつも最適解を求めることができるという性質を残している．また，枝刈りの条件は設定する目的関数や問題の性質により異なるが，今回取り扱う問題での枝刈りの条件については後に述べる．

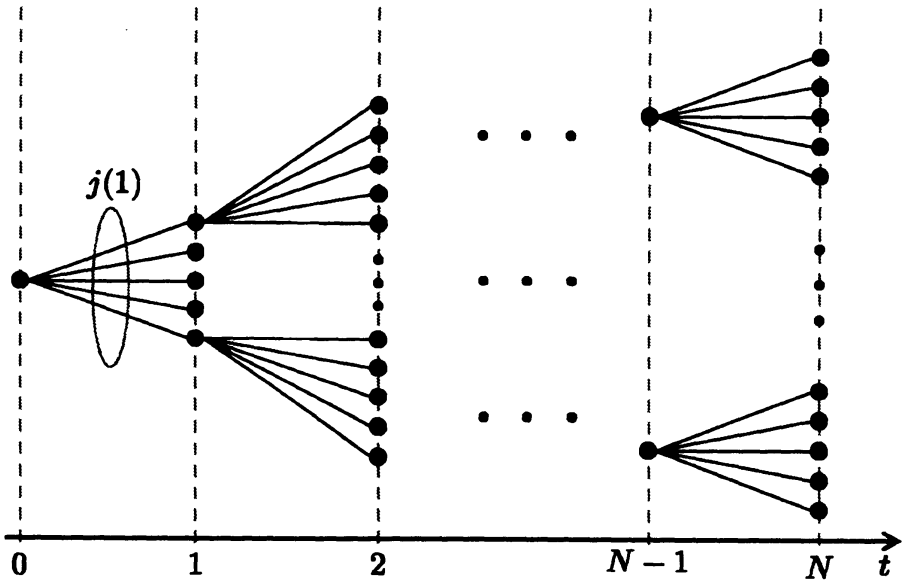

図2: 探索木
(探索木では図の右方向を深さ，上下の広がりを幅と呼ぶ.)

3.2 分枝限定法に基づく近似解法

先ほど述べた分枝限定法の効率は枝刈りをどの程度行なえるかによって大きく異なる．枝刈りにより，探索する枝の数は全探索と比べて少なくなるものの，多くの場合には，依然としてNの指数関数のオーダーの探索を行う必要があり，実用的な時間で解が得られるレベルまでの計算量の削減は見込めない．そこで，ただ分枝限定法をそのまま適用するのではなく，分枝限定法を利用して，近似的な最適解を求めることを考える．まず，これから説明する方法は次のような2つの考察に基づいている．

- 本問題では，期間のはじめの方に運ばれてくる原料の搬入先の決定に期間の後半の方の情報が与える影響は少ないと考えられる．
- 探索する枝の数は期間Nに対して指数関数的に増大していくので，計算時間を短くするためには期間を短くすることが最も効果的である．

この2つの考察をもとに，分枝限定法を全期間に適用するのではなく，短い部分期間に分けて複数回適用するという方法を考えた．ここで,「先読み (Look ahead)」と「実行 (Acceptance)」という2つの考えを導入する．

「先読み」: 短い部分期間に対して分枝限定法を適用して部分的な最適解を得る.
「実行」:「先読み」で得られた部分的な最適解のうちの一部を実際の解として採用する.

また, これ以降「先読み」を行う期間の長さを先読み期間,「実行」を行う期間の長さを実行期間と呼ぶことにする. この「先読み」と「実行」を繰り返し適用することで, 全期間の近似解を得るのが今回提案する近似解法である. 近似解法の求解は図 3 のようなイメージになる.

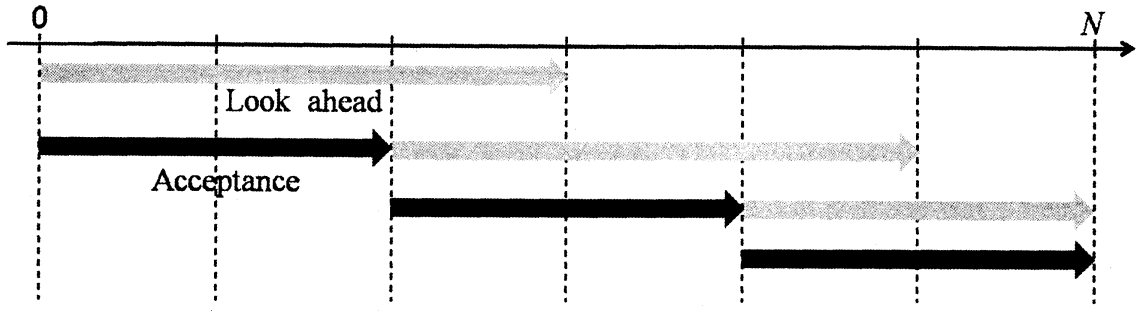

図 3: 近似解法のイメージ

ここで, この解法の中で繰り返し適用することとなる分枝限定法における探索方法および枝刈りの条件について述べる. 探索は, 探索木の深い方向へと単純に探索を進めていく深さ優先探索を用いた. また, 実際に用いた枝刈りの条件は以下の 2 つである.

- 探索途中で充填率が上下限を超えた場合.
- 探索途中での部分目的関数値がそれまでに得られている最も良い解の目的関数値を超えた場合.

1 つ目の条件はそこで解として許容できなくなるため, 2 つ目の条件は今回扱う問題では, 探索を行う中で目的関数値が単調に増加するので, 現在保持しているものより良い解が得られないため探索を打ち切る.

3.3 在庫管理問題への適用

先ほど述べた分枝限定法に基づく近似解法を実際に在庫管理問題に適用する際の概要を以下に示す.

Step1. パラメータの決定
先読み期間と実行期間を決定する

Step2. データの読み込み
倉庫の容量などの必要なデータを読み込む

Step3. 先読み
読み込んだデータを用いて分枝限定法を実行し, 先読み期間分の部分最適解を求める.

Step4. 実行
Step3 で求めた部分最適解を用いて実行期間分の全ての倉庫の充填率を計算していく.

Step5. 先読みと実行の繰り返し
全期間分の部分解が得られるまで先読みと実行を繰り返す.

Step6. 結果の出力
得られた部分解をまとめて全期間の解として出力する.

4 数値実験（1）

ここでは，先ほど提案した近似解法の性能を確かめるために行った数値実験の結果を示す．従来法である動的計画法による解法と計算時間，解の精度の両面において比較を行う．

4.1 実験概要

実験に用いるのは $K = 5$，$N = 90$ の特徴の異なる 4 つの問題例である．解法として従来法である動的計画法と，今回提案した分枝限定法に基づく近似解法を用いる．動的計画法では，全ての倉庫の充填率からなる状態空間の離散化が必要であるが，今回は得られる解の精度と計算時間を考慮し，状態空間を各方向に 80 分割することで離散化を行う．また，提案法では先読み期間と実行期間をパラメータとして設定可能だが，今回は先読み期間を 30 から 60 まで 10 刻みで変化させ，それぞれの先読み期間に対して実行期間を 10 から 10 刻みで先読み期間と同じ長さまで変化させて，以下のような計算機環境で実験を行った．

CPU : Xeon 2.8GHz　Memory : 4GB

4.2 実験結果

4.2.1 問題 1 の結果

先読み期間	実行期間	目的関数値	計算時間（sec.）
30	10	26480.767578	0.15
	20	22765.353516	0.13
	30	23416.287109	0.07
40	10	22765.353516	2.24
	20	26480.767578	0.70
	30	26674.671875	0.49
	40	27451.103516	0.55
50	10	22765.353516	22.67
	20	22765.353516	18.09
	30	22765.353516	8.20
	40	22765.353516	14.77
	50	22765.353516	6.82
60	10	22722.718750	119.11
	20	22765.353516	72.71
	30	22722.718750	47.96
	40	22765.353516	30.06
	50	22765.353516	22.21
	60	22765.353516	21.79
動的計画法		22941.957031	5174.26

この問題は最適解が分かっており，提案法において先読み期間を 60，実行期間を 10，30 に設定した場合には最適解を捉えていた．計算時間の面では提案法が従来法を大きく上回る性能を発揮していることが確認できた．また，目的関数値を見ると提案法では先読み期間，実行期間の設定により非常に精度の悪い解が出力される場合もあるように見えるが，これについても十分に実用可能な解となっている．以下に，最適解

（図 4）および，最も目的関数値の大きくなってしまった提案法において先読み期間 40，実行期間 40 と設定した場合に得られた解（図 5）の充填率の推移のグラフを示す．

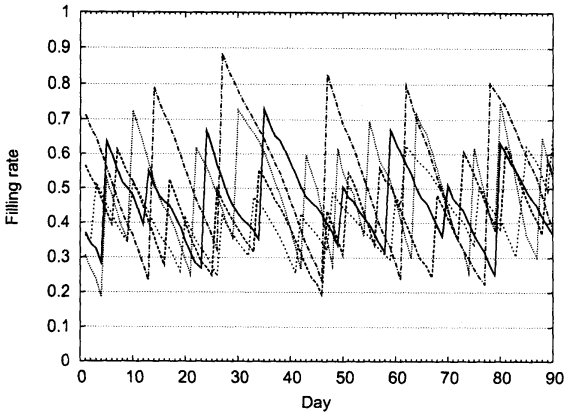

図 4: 最適

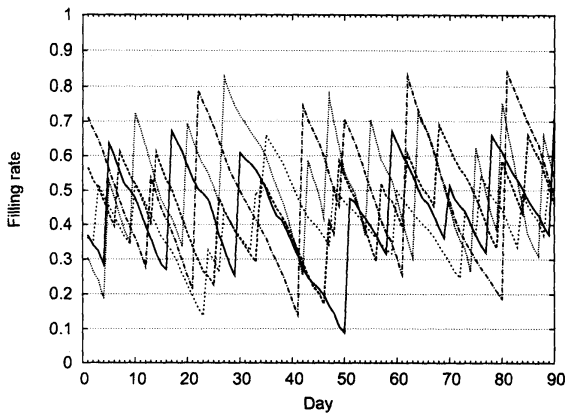

図 5: 先読み期間 40，実行期間 40

4.2.2 問題 2 の結果

先読み期間	実行期間	目的関数値	計算時間（sec.）
30	10	46373.242188	0.58
	20	46042.246094	0.15
	30	47089.207031	0.42
40	10	46407.800781	37.94
	20	46373.242188	33.93
	30	46704.324219	1.95
	40	46470.027344	0.84
50	10	45943.097656	1156.98
	20	45733.398438	157.51
	30	46067.488281	24.67
	40	45616.515625	22.47
	50	48531.980469	18.06
60	10	45719.210938	17281.20
	20	46582.058594	9994.49
	30	45136.566406	9761.69
	40	45943.097656	9538.01
	50	46410.550781	9524.87
	60	46410.550781	9526.43
動的計画法		45430.242188	5109.68

この問題は最適解の分からない問題である．先読み期間，実行期間の設定によっては，提案法により動的計画法と同等以上の解を得られることが確認できた．また，計算時間についてはパラメータの設定によって大きく変化するが，実用的な時間で解を求められることが確認できた．以下に動的計画法によって得られた解（図 6），目的関数値が最も小さくなった先読み期間 60，実行期間 30 と設定した場合の解（図 7），目的関数値が最も大きくなった先読み期間 50，実行期間 50 と設定した場合の解（図 8）のそれぞれの充填率の推移を示す．

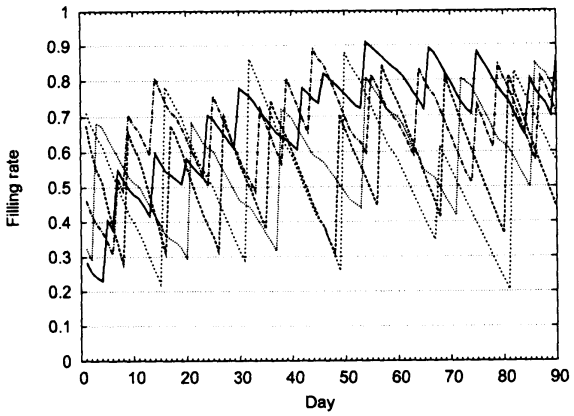

図 6: 動的計画法

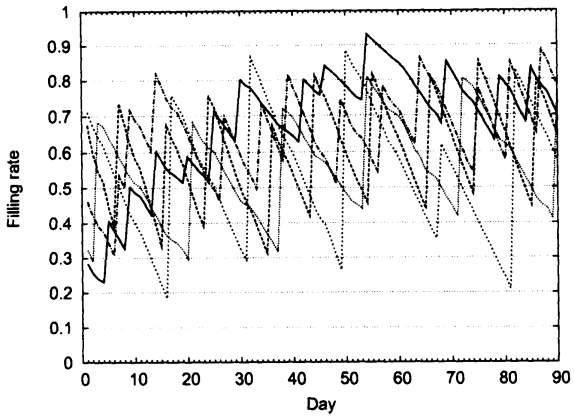

図 7: 先読み期間 60，実行期間 30

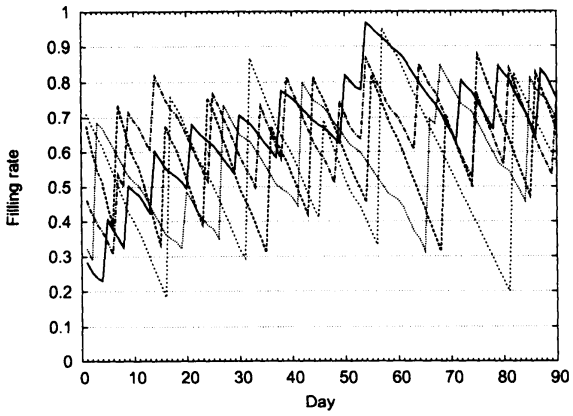

図 8: 先読み期間 50，実行期間 50

4.2.3 問題 3 の結果

先読み期間	実行期間	目的関数値	計算時間（sec.）
30	10	29970.070312	0.85
	20	29486.375000	0.42
	30	28650.300781	0.43
40	10	27382.109375	21.51
	20	29659.275391	12.12
	30	28737.128906	13.92
	40	29742.267578	10.16
50	10	27382.109375	617.75
	20	27382.109375	133.47
	30	27487.935547	462.75
	40	27382.109375	45.81
	50	29480.609375	2.51
60	10	27522.906250	14714.20
	20	27382.109375	6728.13
	30	27382.109375	3233.28
	40	27382.109375	112.14
	50	29480.609375	69.12
	60	-	-
動的計画法		27487.935547	5762.38

この問題も，最適解の分からない問題である．先読み期間，実行期間の設定によっては，提案法により動的計画法と同等以上の解が得られることが確認できた．また，計算時間についてはパラメータの設定によって大きく変化するが，実用的な時間で解を求められることが確認できた．先読み期間 60，実行期間 60 とした場合には，2 度目の部分問題に分枝限定法を適用する際に求解が不可能となった．以下に動的計画法によって得られた解（図 9），目的関数値が最も小さくなった先読み期間 40，実行期間 10 などに設定した場合の解（図 10），目的関数値が最も大きくなった先読み期間 30，実行期間 10 と設定した場合の解（図 11）のそれぞれの充填率の推移を示す．

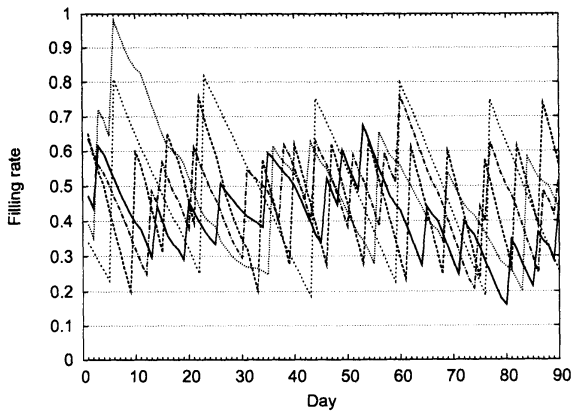

図 9: 動的計画法

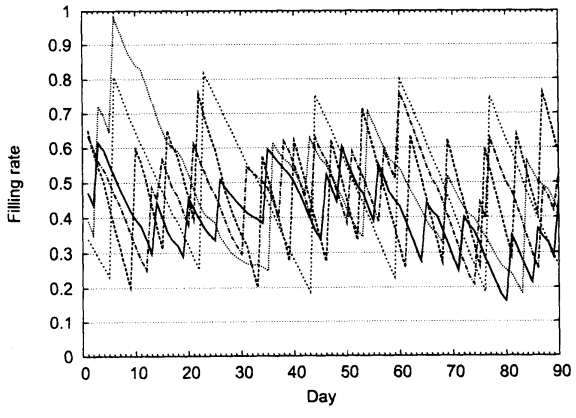

図 10: 最良

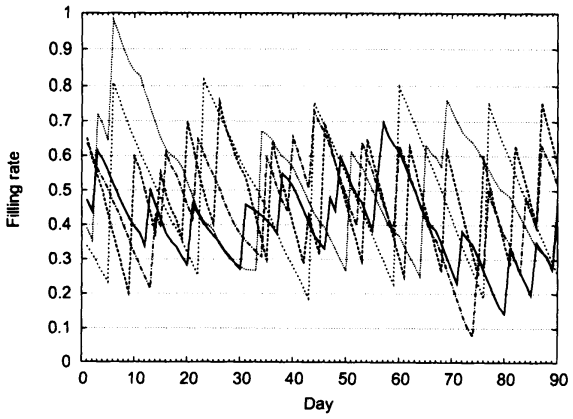

図 11: 先読み期間 30，実行期間 10

4.2.4 問題 4 の結果

先読み期間	実行期間	目的関数値	計算時間（sec.）
30	10	63431.355469	48.93
	20	63302.027344	47.78
	30	64442.964844	47.72
40	10	61991.878906	141.48
	20	63486.503906	134.45
	30	64588.503906	122.22
	40	65222.230469	126.60
50	10	62159.476562	806.16
	20	62159.476562	721.24
	30	62159.476562	700.33
	40	62159.476562	720.00
	50	62159.476562	660.99
60	10	61841.839844	11244.40
	20	61841.839844	10730.10
	30	61894.296875	9856.52
	40	61894.296875	9583.89
	50	61974.730469	9521.32
	60	61974.730469	9517.06
動的計画法		63573.730469	6119.12

この問題も，先の 2 問と同じく最適解の分からない問題である．先読み期間，実行期間の設定によっては，提案法を用いて動的計画法と同等以上の解が得られることが確認できた．計算時間はやはりパラメータの設定によって大きく変化するが，実用的な時間で解を求められることが確認できた．以下に動的計画法によって得られた解（図 12），目的関数値が最も小さくなった先読み期間 60，実行期間 10 などに設定した場合の解（図 13），目的関数値が最も大きくなった先読み期間 40，実行期間 40 と設定した場合の解（図 14）のそれぞれの充填率の推移を示す．

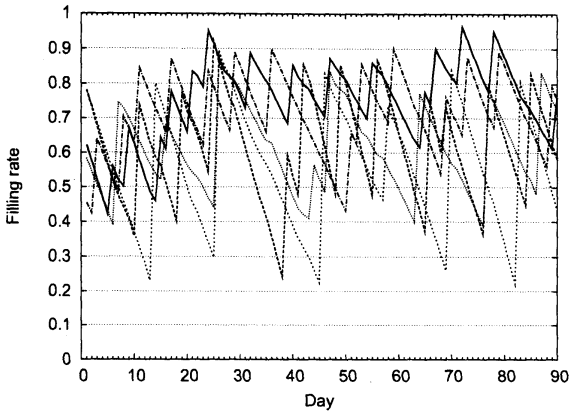

図 12: 動的計画法

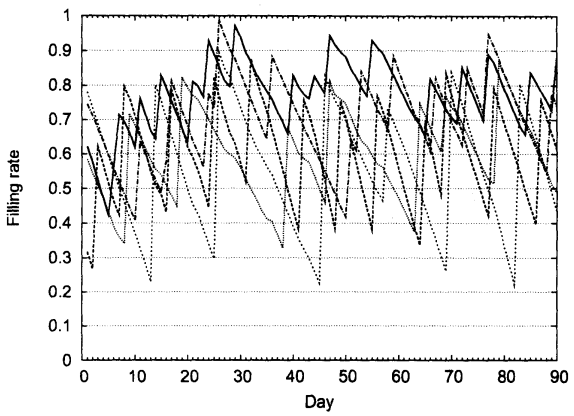

図 13: 最良

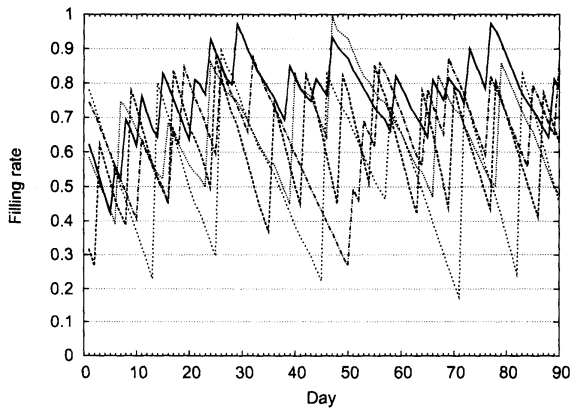

図 14: 先読み期間 40，実行期間 40

4.3 数値実験（1）のまとめ

数値実験により，提案法について次のことが分かった．

- 従来法である動的計画法よりも良い解を得られる可能性がある．
- パラメータによっては実用的な解を高速に求めることが可能である．
- 安定して良い解が得られるように先読み期間を長くすると計算時間が実用的でなくなってしまい，これについて改良が必要である．

5 提案法の改良

前章の実験結果より，提案法は実用のためにはさらなる高速化が必要であると考えられる．提案法において計算時間のほとんどは「先読み」の部分，つまり，部分問題に対して繰り返し適用する分枝限定法の計算が占めている．また，分枝限定法の計算時間は枝刈りの効率によって大きく変化することが知られている．現在，提案法の中で用いる分枝限定法の枝刈りの条件は先にも述べたように以下の２つである．

- 探索途中で充填率が上下限を超えた場合．

- 探索途中で部分目的関数値がそれまでに得られている最小の目的関数値を超えた場合.

このうち，2つ目の条件で枝刈りがどの程度起こるかは，良い解を早期に発見できるかどうかにかかっている．そこで，ここでは良い解を早く発見できるような探索方法を考える．

5.1 探索順序の変更

良い解を早期に発見し，枝刈りがより高頻度で行われることを目指し，次のような探索法を考える．

Step1. ある深さ分だけ探索を行い，その末端に現れる全ての状態についてそれぞれの部分目的関数値を計算・比較する．

Step2. 部分目的関数値の小さいものから先を，先ほどと同じ深さ分の探索，部分目的関数値の計算・比較を行う．

Step3. Step2 を繰り返し，探索木の末端まで探索が終わったら 1 つ前のブロックに戻り，先ほど選択した節点の次に部分目的関数値の小さい節点から先を同様に探索する．

Step4. Step2，Step3 を繰り返し，全ての探索を行う．

この探索方法において，まとめて探索を行う 1 つの部分をブロックと呼びその深さをブロック幅と呼ぶことにする．これを簡単に図で表すと図 15 のようになる．

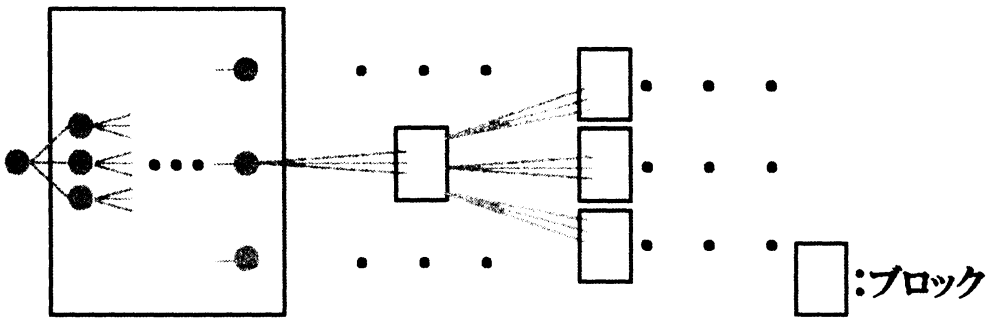

図 15: 新たな探索法のイメージ

この方法では，あるブロックから次に深い位置にあるブロックへと探索を進める際に，貪欲により良い解を求める形になっているため，良い解が早期に発見されることが期待される．

6 数値実験（2）

探索法を改良したことにより，提案法の性能がどのように変化したかを確認する．具体的には，改良を加える前後での計算時間，それぞれの部分問題の最適解を発見するまでに探索した枝の数，および探索した枝の総数をいくつかのパラメータ (先読み期間・実行期間・ブロック幅) について比較する．

6.1 実験概要

実験に用いるのは数値実験（1）で用いた $K = 5$, $N = 90$ の問題例のうち特に計算時間が過大になる傾向がみられる問題2および問題4である．解法として提案法と提案法の探索方法に改良を加えたものの2つを用い，全体の計算時間および，部分問題に対して繰り返し適用する分枝限定法1回ごとの最適解が発見されるまでに探索した枝の数，探索した枝の数についての比較を行う．またパラメータについて今回は比較的短い計算時間で安定して良い解の得られていた先読み期間50，実行期間20と長い計算時間が必要であるが最高レベルの解が得られていた先読み期間60，実行期間10を用いる．改良を加えたものでは新たなパラメータとしてブロック幅を設定可能であるが，今回はこれを10として実験した．計算機環境は数値実験（1）と同じである．

6.2 実験結果

6.2.1 問題2の結果

先読み期間 50	改良前		改良後 (ブロック幅 10)	
実行期間 20	最適解発見	探索総数	最適解発見	探索総数
1〜50 日目	33,854,705	66,344,523	20,987,692	45,243,302
21〜70 日目	450,302,035	576,840,897	4,219,851	444,763,124
41〜90 日目	19,406,382	28,493,957	797,497	17,204,456
計算時間	157.51(sec.)		121.21(sec.)	

先読み期間 60	提案法		改良後 (ブロック幅 10)	
実行期間 10	最適解発見	探索総数	最適解発見	探索総数
1〜60 日目	17,444,884,808	37,750,584,132	11,959,868,955	29,602,458,540
11〜70 日目	17,466,234,176	30,997,891,979	1,313,033,969	24,528,921,789
21〜80 日目	710,025,300	1,892,272,220	92,796,975	1,519,211,932
31〜90 日目	903,668,052	1,091,560,716	8,148,436	625,632,387
計算時間	17281.20(sec.)		14313.90(sec.)	

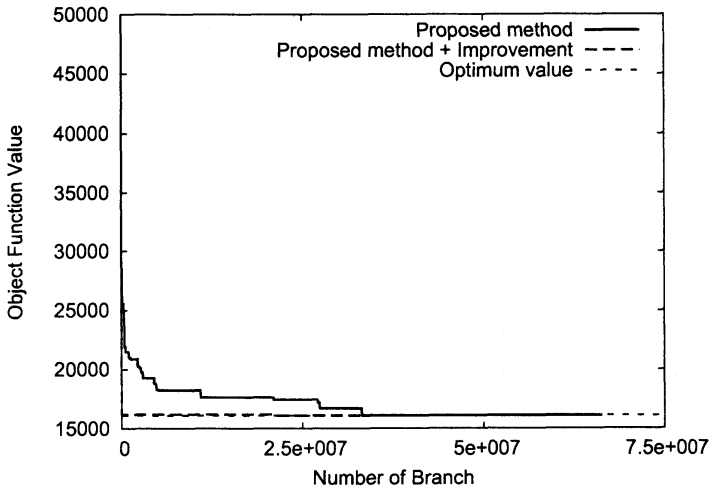

図 16: 暫定解の更新の様子

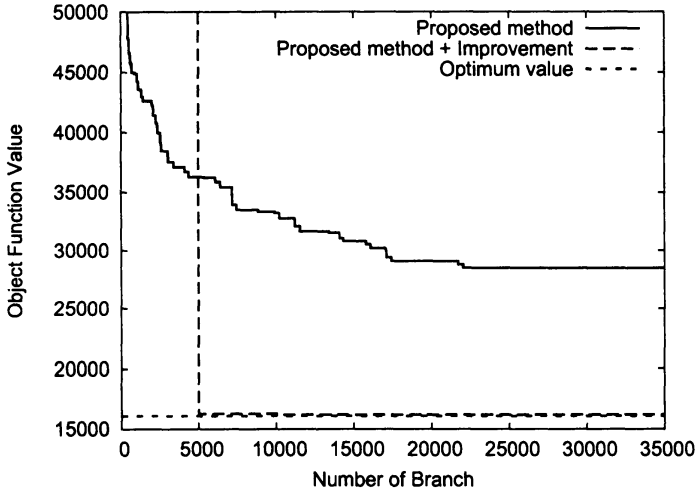

図 17: 序盤の様子

パラメータを先読み期間 50, 実行期間 20 としたものについて改良前，後それぞれについて 1〜50 日目の探索を行った際の，探索した枝の本数と目的関数値の減少の関係を確認してみると，その振る舞いは図 16 のようであった．(探索開始時には目的関数値を適当な大きな値として 100000000.0 を保持している．) また，探索序盤の部分に特に注目してみると図 17 のようになっていた．この図 17 において改良を加えたもののラインが枝の本数 5000 付近で垂直に落ちてういるのは，ここまでは実行可能解が見つからなかったためである．この図では分かりにくいが，先の表を見てもわかるようにこのとき得られた解が最適解ではなく，この後もわずかに目的関数値が減少している．これらの結果より，提案法に加えた改良は良い解を早期に発見するという目的を十分に達成していることが分かる．しかし，良い解が見つかってから先も多くの枝を探索しており，探索した枝の総数を大きく減らすには至らなかったことから, 枝刈りの条件の適切さに疑問が残る結果となった．

6.2.2 問題 4 の結果

先読み期間 50	改良前		改良後 (ブロック幅 10)	
実行期間 20	最適解発見	探索総数	最適解発見	探索総数
1〜50 日目	815,640,206	2,795,480,007	199,765,854	2,529,820,353
21〜70 日目	4,933,187	5,641,053	352,563	2,927,025
41〜90 日目	169,165,059	283,958,232	16,441	215,449,858
計算時間	720.24(sec.)		655.81(sec.)	

先読み期間 60	提案法		改良後 (ブロック幅 10)	
実行期間 10	最適解発見	探索総数	最適解発見	探索総数
1〜60 日目	12,832,264,625	43,522,770,186	2,574,162,466	39,282,537,976
11〜70 日目	542,556,467	1,039,392,041	59,800,775	732,439,382
21〜80 日目	3,411,183,519	4,945,629,065	5,021,626	2,810,179,266
31〜90 日目	835,861,564	1,350,074,764	36,576,276	908,999,675
計算時間	11244.40(sec.)		9815.83(sec.)	

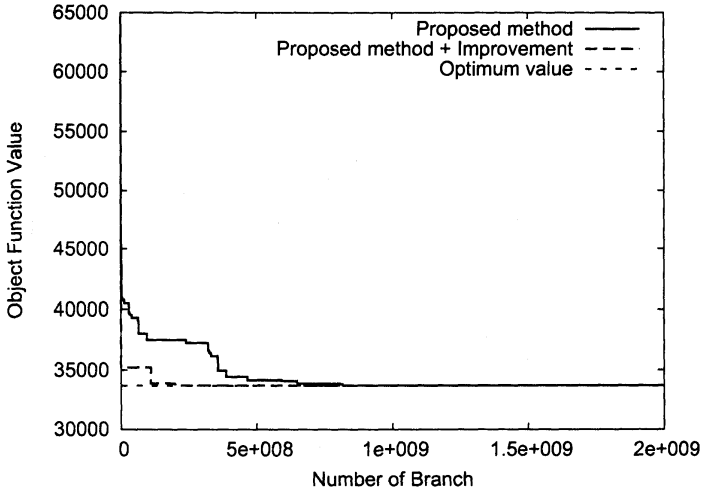

図 18: 暫定解の更新の様子

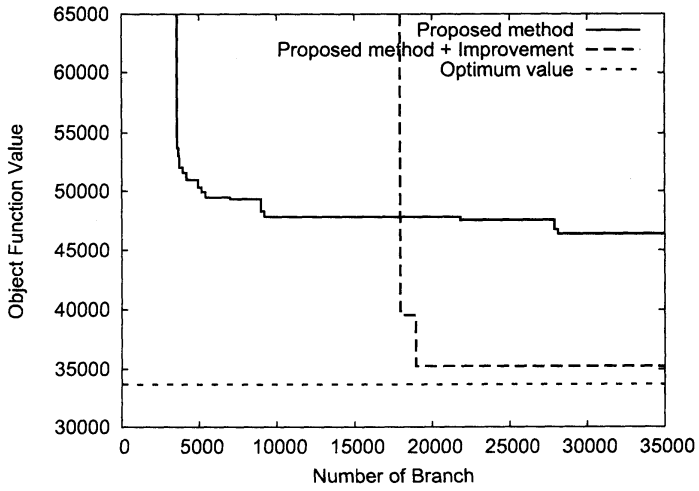

図 19: 序盤の更新の様子

パラメータを先読み期間 50, 実行期間 20 としたものについて改良前，後それぞれについて 1〜50 日目の探索を行った際の，探索した枝の本数と目的関数値の減少の関係を確認してみると，その振る舞いは図 18 のようであった.（まだ後半にここでは表示しきれない部分がある.）また，探索序盤の部分での振る舞いは図 19 のようになっていた．こちらの問題の結果からも，先の問題 2 の結果において述べたことと同じ考察を得ることができる.

7 おわりに

7.1 まとめ

今回は，本稿において次のことを述べた．

- 我々が研究の対象としている在庫管理問題を紹介し，この問題に対して分枝限定法に基づく近似解法を提案した．また，提案法と従来法である動的計画法の性能比較を行い，提案法の改良の必要を確認した．
- 提案法の中で繰り返し用いる分枝限定法の探索方法の変更するという改良を考え，この効果を確認したところ枝刈りの条件についての問題点を発見した．

7.2 今後の課題

- 今回把握した問題点である枝刈りの条件について，問題の性質に合わせた新たな条件を提案する．
- 他のヒューリスティック解法との比較を行う．

参考文献

[1] 李天ら: ある在庫管理問題の動的計画法による解法と CUDA を用いた高速化，情報処理学会論文誌 ACS，Vol. 1，No. 2，pp. 1-10 (2008).

[2] M.R.Garey, D.S.Johonson.: COMPUTERS AND INTRACTABILITY, FREEMAN Publications(1979).