Vol.2017-MPS-116 No.5
2017/12/11

初期集団の改良によりパレートフロントへの収束性を高めた多目的遺伝的アルゴリズムによるIT プロジェクトスケジューリング

小林 敬明 [1,2,a)]　森口 聡子 [1]

概要：本研究の目的は，IT プロジェクトにおいてプロジェクトの再計画が発生した場合，計算機でプロジェクトのスケジュールを自動的に生成することにより，スケジュールの最適化，再作成工数の削減，再作成の迅速化を実現することである．一般的にプロジェクトの納期と予算はトレードオフの関係にあり，優先度はプロジェクトがおかれた環境や状況次第で変化する．そこで本研究では，(1)納期，(2)要員の重複タスク日数，(3)要員数の 3 式の目的関数を最小化するための多目的遺伝的アルゴリズムを用いたスケジュール生成ソフトウェアを提案する．このソフトウェアは，遺伝的アルゴリズムで使用する初期集団にパレートフロントの端に位置する複数のパレート最適解を予め含めることにより，パレートフロントへの収束性を高めている．ユーザは，このソフトウェアで生成された複数の準最適解の中から，プロジェクトの状況にあったスケジュールを選択することができる．

キーワード：プロジェクトスケジューリング,遺伝的アルゴリズム,多目的最適化

IT project scheduling based on a multi-objective genetic algorithm with fast convergence to Pareto front by an improved initial population strategy

TAKAAKI KOBAYASHI[1,2,a)]　SATOKO MORIGUCHI[1]

Abstract: The purpose of this research is to realize an automatic generation of a project schedule which optimizes the schedule, reduces the number of rebuild man-hours and speeds up the re-creation when the project re-planning occurs in the IT project. In general, the project due date and the price have a trade-off relationship, and the priority varies depending on the environment and situation of the project. Therefore, in this research, we propose schedule generating software using a multi-objective genetic algorithm to minimize objective functions of (1) the due date, (2) the number of days in duplicate task for a member, and (3) the number of members. This software enhances the convergence speed to the Pareto front by a strategy in which an initial population of genetic algorithm includes beforehand some Pareto optimal solutions located at the edge of the Pareto front. The user can select a schedule suitable for the situation of the project from some semi-optimal solutions generated by this software.

Keywords: project scheduling, genetic algorithm, multi-objective optimization

1. はじめに

1.1 研究の背景

日経コンピュータの調査[1]によると, IT 業界におけるプロジェクトの平均成功率は 31.1%とのことである．ここでの成功の定義はスケジュール超過，コスト超過や，プロジェクト中止を経ずに当初計画通りに完遂できたものを示す．

スケジュール超過の原因の内訳を見ると，要件定義が長くなった（43.6%），設計作業が長くなった（33.0%），開発作業が長くなった（33.0%）と，プロジェクト実行段階において当初計画より長い期間がかかったことが上位を占めている．また，コスト超過の原因の内訳を見ると，追加の開発作業が発生（58.9%），追加の設計作業が発生（47.5%），追加の企画作業が発生（32.6%）と，当初の計画になかった追加作業がプロジェクト実行段階に発生したことが上位を占めている．

このように IT プロジェクトにおいては, 計画通り遂行を阻害する事象が多発している. 計画に変更が発生した場合, 再計画を迅速に行わなければならないが，ほとんどの現場において経験と勘と度胸に頼る手動での再計画作業が行なわれているため[2], スケジュール再作成の属人的な能力への依存と効率の悪さがプロジェクトマネジメントを行う上での課題となっている．

1.2 先行技術と提案するソフトウェア

プロジェクトマネジメントの効率向上策として，プロジェクトマネジメントツールの導入が挙げられるが，Microsoft Project など市販のプロジェクトマネジメントツールには, 納期と IT プロジェクトのコストの主要素である要員数の両方の最小化に対するトレードオフを考慮した複

1 首都大学東京大学院社会科学研究科経営学専攻
Department of Business Administration, Graduate School of Social Sciences, Tokyo Metropolitan University
1-1 Minami-Osawa, Hachioji-shi, Tokyo, Japan
2 株式会社 TransRecog
TransRecog CO., LTD.
a) takobaya@transrecog.com

数の最適解を提案する機能はない.

そこで，プロジェクトの納期とコストのトレードオフを考慮した IT プロジェクト向けの自動スケジューリングソフトウェアを提案する．提案するソフトウェアは，予め制約条件を入力すると，その制約条件を満たす複数のパレート準最適解をデータとして出力するものとし，またそのデータをユーザがガントチャートとして閲覧することにより準最適解の集合の中からプロジェクトの状況にあったスケジュールを選ぶことができるものとする．ユーザは，必要に応じて，選択した準最適解を Excel やプロジェクトマネジメントツールにインポートして，プロジェクトマネジメントに使用できるものとする．

提案するソフトウェアの対象 IT プロジェクト規模は，日本情報システム・ユーザー協会が大別する 100 人月未満，100 人～500 人月，500 人月以上のうち，100 人月未満の中央値である 50 人月を対象とする．一般的に 1 タスクあたり 2 週間が目途である[3]ため，50 人月をタスク数に変換すると，50 人月×20 日（1 か月の稼働日数）÷10 日（2 週間稼働日数）=100 タスクとなる．

1.3 先行研究

プロジェクトスケジューリング問題の中で，優先順位制約やリソース制約など資源の制限を課した問題を「資源制約付きプロジェクトスケジューリング問題」という．プロジェクトスケジューリングを行う際の代表的な問題として，時間とコストのトレードオフがある．これは「離散時間コストトレードオフ問題」（Discrete time-cost tradeoff problem; DTCTP）と呼ばれ，近年研究が盛んにおこなわれている．DTCTP の解法は，厳密解アルゴリズム，ヒューリスティクスアルゴリズム，メタヒューリスティクスアルゴリズムの 3 種類に大別される[4]. DTCTP は業種を特定しない汎用的なプロジェクトスケジューリングに関する研究が主流であるが，IT プロジェクトは一般的なプロジェクトと比べ，パラメータや目的関数（評価関数）が異なると考える．IT プロジェクトの特徴として，知識労働集約型である点，要員の能力差が大きい点，要員の属性に応じて配置が決まる点があげられる．

2. 目的関数と制約条件

2.1 目的関数

以上の IT プロジェクトの特徴から，

(1) プロジェクト完了日数
(2) 全要員の重複タスクの日数の合計
(3) プロジェクトに参画する要員数

を最小化することを目的とする．

(1) は DTCTP で主題となる時間を表す目的関数，(3)は同じく DTCTP で主題となるコストについて示したものである．IT プロジェクトは知識労働集約型業務であり，コストは人件費が主体となるため，コストを示す目的関数をプロジェクトに参画する要員数とする．(2)は，IT プロジェクトは要員の能力差が大きい点，プロジェクトの状況から要員の好み・健康状態に応じて要員の配置を決めることがある点から，要員に複数のタスクを並行して実施させることが十分に考えられることを考慮した，全要員の重複タスクの日数の合計を表す目的関数である．

2.2 制約条件

本研究では，IT プロジェクトのスケジューリングで一般的に用いられている以下を制約条件として設定する．

(a) プロジェクトスケジュールは複数のタスクから構成され，各タスクには先行タスクと期間（日数）が設定されている．
(b) 最初のタスクを除く全てのタスクには先行タスクが設定されている．
(c) 先行タスクは複数設定できる．
(d) タスクに割当てる要員は要員リストに登録されている．
(e) 1 タスク割当てる要員数は 1 名である．
(f) プロジェクトスケジュールは 1 つのタスクから始まり，1 つのタスクで終わる．

最小化したい関数が 3 つであることから，多目的最適化問題となる．多目的最適化問題におけるパレート最適解集合を獲得する手段として，進化計算が注目されている[5]. 進化計算による多目的最適化は，産業応用との親和性も高い[5]. そこで本研究では，進化計算による多目的最適化で最も有名なアルゴリズムとして応用分野で頻繁に利用されている[5], NSGA-II(Fast Elitist Non-dominated Sorting Genetic Algorithm-II)[6]を採用することとする．

3. 多目的最適化によるモデルと数値実験

3.1 制約条件の設計

制約条件を表すため，以下のデータ構造で表す.

$(k_1, d_1, \boldsymbol{A}_1, taskname_1)$,
$(k_2, d_2, \boldsymbol{A}_2, taskname_2)$
,…,
$(k_n, d_n, \boldsymbol{A}_n, taskname_n)$

ここで

k_i：タスク番号
d_i：タスクの所要日数
$\boldsymbol{A}_i = (a_{i1}, a_{i2}, \dots, a_{ij_i})$
　但しa_{ij_i}は先行タスク番号.
　j_iは該当タスクに対応する先行タスクの数で，
　先行タスクがない場合は$a_{i1} = 999999$
$taskname_i$：タスク名（文字列）

$i = 1,2,\dots, n$ (nはプロジェクトの総タスク数)

である．

制約条件の中に出てくる要員リストは，以下のデータ構造で表す．

$(e_1, membername_1),$
$(e_2, membername_2)$
$,\dots,$
$(e_n, membername_n)$

ここで

e_i：要員番号

$membername_i$：要員名（文字列）

$i = 1,2,\dots, n$ (nは要員数)

である．

3.2 遺伝子の設計

遺伝子は以下のデータ構造とする．

$(t_1, m_1), (t_2, m_2), \dots, (t_n, m_n)$

ここで

t_i：タスク開始遅延日数

m_i：要員番号

$=$ (要員リスト$e_1, \dots, e_j$のうちいずれか 1 つ)

$i = 1,2,\dots, n$ (nはプロジェクトの総タスク数)

$j = 1,2,\dots, c$ (cは要員数)

である．

上記のように設計することで，制約条件を守った状態で 3 つの目的関数，(1)プロジェクト完了日数，(2)全要員の重複タスクの日数の合計，(3)プロジェクトに参画する要員数のうちいずれか 2 つを最小化するパターンを生成できる．

3.3 評価関数

評価関数は制約条件と遺伝子から目的関数を生成する関数である．処理は以下の順に行う．

(a) トポロジカルソート結果を用いた各タスク開始終了日の算出

制約条件にある各タスクに対応する先行タスクのデータを用いたグラフを内部的に生成し，トポロジカルソートにてシリアライズを行う．制約条件より，グラフは有向非巡回グラフとなるので，トポロジカルソートが可能となる．なおトポロジカルソートは，1 回だけ行えばよいので，遺伝的アルゴリズムの本処理を行う前に予め実行しておく．

シリアライズ後のタスクデータを用いて，先頭から順番にタスクの開始終了日を算出する．タスクの開始日は，終了日算出済みの先行関係のあるタスクの中から，最も遅いタスクの終了日の翌日に，対応する遺伝子のタスク開始遅延日数を加算した値とする．併せて対応する遺伝子の要員番号に従い要員をタスクに割当てる．

(b) プロジェクト完了日数の算出

生成した各タスク開始終了日の中から最も遅いタスク終了日を目的関数(1)プロジェクト完了日数とする．

(c) プロジェクトに参画する要員数の算出

割当てられた要員数を目的関数(2)プロジェクトに参画する要員数とする．

(d) 全要員の重複タスクの日数の合計の算出

要員毎に，一つの日に複数のタスクが割当てられている日について，重複しているタスク数-1 を順に積算してゆく．例えばある要員がある日に 3 タスク重複している場合は，2 を積算する．この値を目的関数(3)全要員の重複タスクの日数の合計とする．

3.4 提案するソフトウェアの流れ

提案するソフトウェア全体の流れを図 3-1 に示す．入力は制約条件と要員リストであり，出力はパレート準最適解を表す遺伝子とガントチャートのデータである．

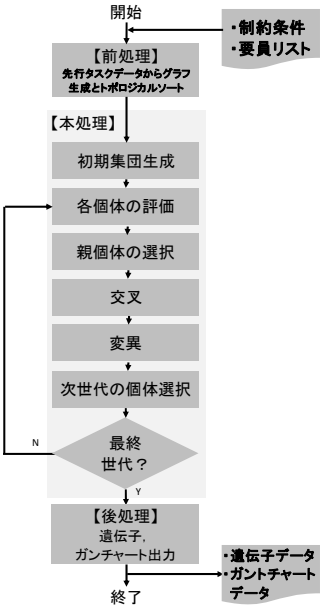

図 3-1　提案するソフトウェア全体の流れ

Fig.3-1 Flowchart of the proposed software.

3.5 提案するソフトウェアの環境及び構成

提案するソフトウェアは，実務での利用を考慮しプログラミング実行環境として急速に利用が拡大している Python で動作するように開発する．提案するソフトウェアが使用するライブラリと使用用途を表 3-1 に示す．いずれも GNU General Public License もしくは BSD ライセンスであり，無償で利用できる．商用ソルバも使用しないため，OS 以外にソフトウェア購入費用はかからず，容易に実務で利用できる．

表 3-1　ライブラリと利用用途

Table 3-1 Library and usage.

ライブラリ	利用用途
NetworkX[7]	グラフの生成とトポロジカルソート
Python-Gantt[8]	ガントチャートデータ生成
NumPy[9]	各種数値計算
DEAP[10]	遺伝的アルゴリズム

3.6 数値実験 1

提案するソフトウェアの性能と精度を図る目的として 10 タスクの制約条件と，10 人の要員リストにて数値実験を行う．実験環境を表 3-2 に示す．遺伝的アルゴリズムの世代数は 200，個体数は 200 とする．

表 3-2　実験環境

Table 3-2 Experiment environment.

区分	項目	内容
ハードウェア	CPU	Intel Core i5-4300 (1.90GHz)
	メモリ	DDR3L SDRAM 12GB (800MHz)
ソフトウェア	OS	Windows 7(64bit)
	実行環境	Python 3.6.1
	ライブラリ	NetworkX 1.11
		Python-Gantt 2.8.0
		NumPy 1.12.1
		DEAP 1.1.0

目的関数が 3 つであるため，実験結果を 3 次元の散布図（図 3-2）に示す．図中の吹き出しは代表的な解を示す．

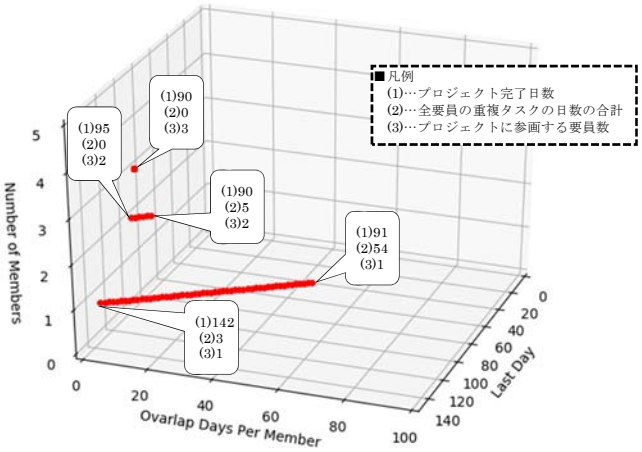

図 3-2　実験結果（数値実験 1）

Fig.3-2 Experimental results (Numerical experiment 1).

全ての解がパレートフロント上にあることを確認できた．計算時間は，プログラム開始から 200 世代目計算完了までが 1 分 16 秒であった．

3.7 数値実験 2

続いて 100 タスクの制約条件と，50 人の要員リストにて数値実験を行う．制約条件は実務で使用されたスケジュールデータから 100 タスク分を抜粋したものを使用する．実験環境は数値実験 1 の表 3-2 と同じである．遺伝的アルゴリズムの世代数は 300，個体数は 200 とする．

実験結果を図 3-3 に示す．計算時間は，プログラム開始から 300 世代目計算完了までが 8 分 5 秒であった．

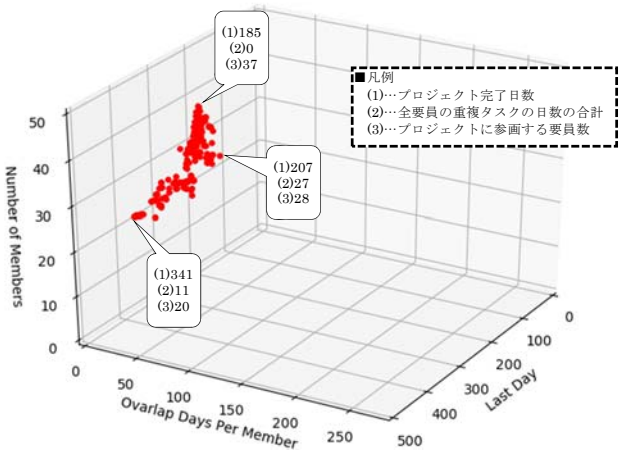

図 3-3　実験結果（数値実験 2）

Fig.3-3 Experimental results (Numerical experiment 2).

この実験結果には数値実験 1 の実験結果のような解集合の広がり（多様性）が見られない．遺伝的アルゴリズムはヒューリスティクスアルゴリズムであり，必ずしも真のパレートフロントの求解はできない点を考慮しても，目的関数(3)プロジェクトに参画する要員数は最小 20，最大 37 と範囲が狭い．

本研究のモデルから考慮すると，3 つの目的関数，(1)プロジェクト完了日数，(2)全要員の重複タスクの日数の合計，(3)プロジェクトに参画する要員数のうち，(2)(3)を最小にした状態で(1)を最小化する遺伝子，(1)(3) を最小にした状態で(2)最小化する遺伝子は，遺伝的アルゴリズムを用いなくても容易に計算できる．そこで，(1)(2) を最小にした状態で(3)を最小化する遺伝子と併せた計 3 つのパレートフロントの端に位置する遺伝子について，遺伝的アルゴリズムを実行する前に算出し，初期集団に含めることとする．これらを含めることにより，解集合の広がり（多様性）を期待できると考える．また，3 つの遺伝子はいずれもパレート最適解であるため，交叉や変異で書き換えられないようにブロックすることで，最終世代まで生き残るようにする．本研究では，これら 3 つの遺伝子を「保存遺伝子」と名付けることとする．

4. 改良したモデルを用いた提案ソフトウェア

改良したモデルを用いた提案ソフトウェアの全体の流れを図 4-1 に示す．改良前（図 3-1）から追加した箇所は赤枠で記す．

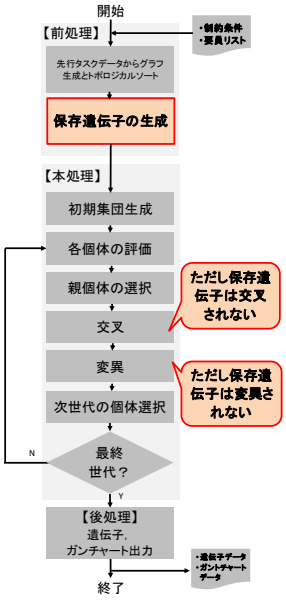

図 4-1　改良提案ソフトウェアの全体の流れ
Fig.4-1 Flowchart of our improved software.

4.1 保存遺伝子の生成アルゴリズム

(a) 目的関数(2)全要員の重複タスクの日数の合計，目的関数(3)プロジェクトに参画する要員数が最小の状態で目的関数(1)プロジェクト完了日数を最小にする遺伝子

これは，トポロジカルソートされたタスク順に重複しないようタスクを左詰めに並べることで容易に算出できる．要員は，要員リストからランダムに選んだ 1 名を全てのタスクに割当てる．目的関数 (2)全要員の重複タスクの日数の合計は 0 日，目的関数(3)プロジェクトに参画する要員数は 1 人となる．

(b) 目的関数(1)プロジェクト完了日数，目的関数(3)プロジェクトに参画する要員数が最小の状態で目的関数(2)全要員の重複タスクの日数の合計を最小にする遺伝子

これは，トポロジカルソートされたタスク順に重複を許しつつタスクを左詰めに並べることで容易に算出できる．要員は，要員リストからランダムに選んだ 1 名を全てのタスクに割当てる．目的関数(3)プロジェクトに参画する要員数は 1 人となる．

(c) 目的関数(1)プロジェクト完了日数，目的関数(2)全要員の重複タスクの日数の合計が最小の状態で目的関数(3)プロジェクトに参画する要員数を最小にする遺伝子

これは，目的関数(1)プロジェクト完了日数を(b)より得た値に設定し，目的関数(2)全要員の重複タスクの日数の合計を 0 にした場合に，目的関数(3) プロジェクトに参画する要員数を最小化する 1 目的最適化問題となる．求解には，二分探索と分枝限定法を併用することとする．アルゴリズムを以下に示す．

Step1: 要員リストに載っている全要員の数で，全要員の重複タスクの日数が最小となる要員の組合せを分枝限定法にて求める．

Step2: Step1 で得た全要員の重複タスクの日数が 0 より大きいならば，解は要員リストに載っている全要員の数となり，計算を終了する．そうでないならば，

lowMemNum=0

highMemNum

=要員リストに載っている全要員の数

とする．

Step3: centerMemNum=(lowMemNum+highMemNum)/2 とする．centerMemNum の要員数で，全要員の重複タスクの日数が最小となる要員の組合せを分枝限定法にて求める．

Step4: もし highMemNum≤lowMemNum かつ，全要員の重複タスクの日数が 0 より大きいならば

lowMemNum = centerMemNum + 1

highMemNum = centerMemNum + 1

とする．もし highMemNum≤lowMemNum かつ，全要員の重複タスクの日数が 0 ならば，解は centerMemNum となり，計算を終了する．

Step5: もし全要員の重複タスクの日数が 0 より大きいならば，

lowMemNum = centerMemNum + 1

とする．そうでないならば，

highMemNum = centerMemNum – 1

とする．

Step6: Step3 に戻る．

分枝限定法は，全要員の重複タスクの日数が一定期間以上経過しても改善しない場合，打ち切るようにする．

4.2 数値実験 3

数値実験 2 と同じ制約条件，実験環境で改良したモデルを用いた提案ソフトウェアを実験する．実験結果を図 4-2 に示す．

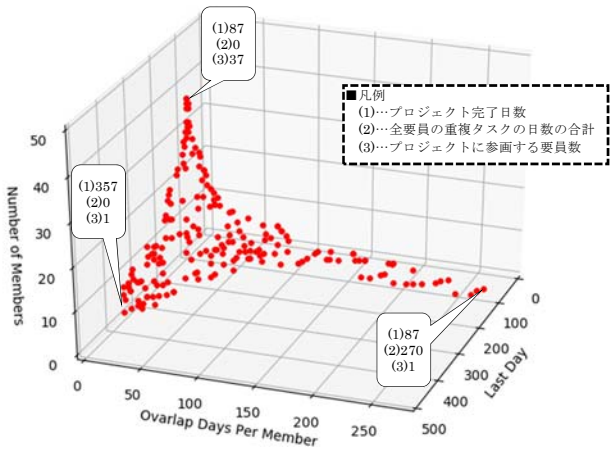

図 4-2　実験結果（数値実験 3）

Fig.4-2 Experimental results (Numerical experiment 3).

改良の企図通り，図 4-2 中の吹き出しの 3 つの保存遺伝子が，初期集団からそのままパレートフロントの端の解として最終世代まで残った．数値実験 2 の結果である図 3-3 と比較して，解集合の広がり（多様性）が大幅に改善された．

実験結果 2 と実験結果 3 を評価するため，それぞれの世代毎の HyperVolume 値を比較する（図 4-3）．HyperVolume の参照点は(10000,10000,100)とした．

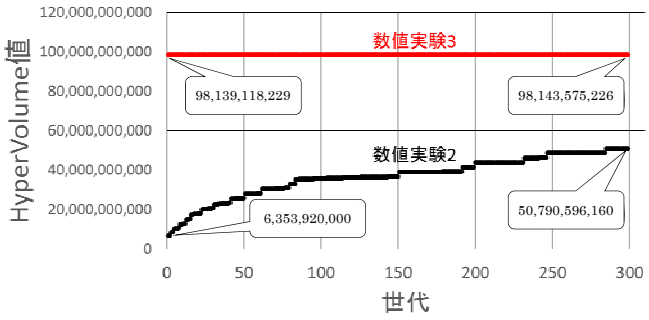

図 4-3　HyperVolume 値比較

Fig.4-3 Comparison of HyperVolume values.

数値実験 3 は数値実験 2 と比べ高い HyperVolume 値を示しているため，改良が解集合の広がり（多様性）に対して有効であったことがわかる．

5. おわりに

本研究では，プロジェクトの納期とコストのトレードオフを考慮した IT プロジェクト向けの自動スケジューリングソフトウェアを提案した．提案するソフトウェアで数値実験をしたところ，100 タスクで解集合について広がり（多様性）が見られなかったため，アルゴリズムを改良し，パレートフロントの端の解を予め計算して初期集団に含めることとした．解集合の広がり（多様性）を確保したため，結果的にパレートフロントへの収束性を高められた．その結果，改良したソフトウェアは，出力した複数の準最適解の中から，ユーザがプロジェクトの状況にあったスケジュールを選択できるようになった．

実験結果は，例えば最短スケジュールで納期に間に合わせる必要がある場合など，プロジェクトの置かれた状況次第では選んだ解をそのままスケジュールとして採用できることを示している．もし提案するソフトウェアで出力したスケジュールについて，細部を調整する必要がある場合でも，スケジュール作成のたたき台として大幅な作成時間短縮が実現できるので，実務に有用である．また実験環境のハードウェアは，2014 年時点の標準的なビジネス用ノート PC であるため，提案するソフトウェアは性能的に実務で十分に活用可能である．

謝辞　本研究は JSPS 科研費 26350430 の助成を受けたものである．

参考文献

[1] 日経 BP 社：第 2 回 プロジェクト実態調査 ,日経コンピュータ,2008 年 12 月 1 日号, pp.38-49 (2008).
[2] 布川 薫：重要性増すモダン PM　そのシステム開発での実践体系を知る,日経 IT プロフェッショナル，2002 年 7 月号, pp.152-157 (2002).
[3] 初田 賢司，神子 秀雄：思い込みや楽観的な予測を排除緻密な WBS 定義こそが重要−作業計画とスケジュール作成の実践知識特集 1　進ちょく管理の大本命　EVM を極第 2 部　作業計画とスケジュール作成の実践，日経 IT プロフェッショナル, 2004 年 12 月号, pp.42-47 (2004).
[4] Madjid Tavana, Amir-Reza Abtahi , Kaveh Khalili-Damghani : A new multi-objective multi-mode model for solving preemptive time–cost–quality trade-off project scheduling problems,　Expert Systems with Applications, Vol.41, pp.1830-1846 (2014).
[5] 佐藤 寛之,石渕 久生：進化型多数目的最適化の現状と課題，オペレーションズ・リサーチ: 経営の科学, Vol.62, No.3, pp.156-163 (2017).
[6] K Deb et al, A Fast and Elitist Multiobjective Genetic Algorithm NSGA-II, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, Vol.6, No.1, pp.182-197 (2002).
[7] Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart: Exploring network structure, dynamics, and function using NetworkX, in Proceedings of the 7th Python in Science Conference (SciPy2008), pp.11–15 (2008).
[8] Alexandre Norman: Python-Gantt, http://xael.org/pages/python-gantt-en.html (2014).
[9] Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux: The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering, Vol.13, pp.22-30 (2011).
[10] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau and Christian Gagné: DEAP: Evolutionary Algorithms Made Easy, Journal of Machine Learning Research, Vol.13, pp.2171-2175 (2012).