Vol.2020-GN-109 No.1
Vol.2020-CDS-27 No.1
Vol.2020-DCC-24 No.1
2020/1/23

Android 端末におけるプリインストールアプリに着目した資源利用最適化手法

清原良三[1] 岩井隆真[1]

キーワード：スマートフォン，Android，最適化，プリインストール，バッチファイル，ADB

概要:現在広く普及しているスマートフォンは目まぐるしく発展しているが,未だに数多くの改善の余地がある．その中の一つにスマートフォン内の限られた資源を有効活用するということが必要である．特にバッテリの持ちの悪化や,メモリ使用量が増えることによって起こる不安定な動作などによってユーザビリティが低下している点が問題である．また,日々スマートフォンを使用する中で通信事業者と契約を結びデータ通信を行っているが,その通信事業者は多様化しており，近年では、格安 SIM と呼ばれることが多い MVNO(仮想移動体通信事業者)の普及が進んでいる．従来の MNO(移動体通信事業者，以下キャリアと呼ぶ)契約時に購入したスマートフォンの SIM カードを差し替えるだけで MVNO へ乗り換えることが可能であるが，そのスマートフォンのパフォーマンスに不満を持っているユーザは買い替えを余儀なくされる．また，キャリアで購入したスマートフォンにはキャリア独自のプリインストールアプリが存在し,ユーザがそれを使用していないにも関わらずユーザの知らない間に限られた資源を使用している場合がある．それらのプリインストールアプリの存在を認識していない場合があり，削除するために端末を初期化しても OS に組み込まれているため削除できず，Android の開発用コマンドを使用したアプリの削除は専門知識を有していない一般ユーザには困難である．以上の問題点を解決するため，本論文では Android 端末を対象に一般ユーザ-でもキャリア製プリインストールアプリを削除できる Windows アプリケーションを開発し，Android 端末として，バッテリーの寿命および操作性という観点から評価した．

1. はじめに

現在スマートフォンは広く普及しており，人々の生活に欠かせない存在となっている．中学生からスマートフォンを所有し友達とのコミュニケーションをとっており，スマートフォン依存症が問題視されるほど人々は常時アクセスする癖がついている．調査によると，2017 年時点で世帯保有率は 75.1%と 72.5%のパソコンを上回っており，個人の保有状況は 60.9%と同じく高い普及率を示している[1].

数十年前に広く利用されていたフィーチャーフォンに比べ，タッチパネルを使用することにより直感的な操作が可能となったが，ユーザの不満というものは未だ存在する．その中でも，購入当初は快適に使用できていたスマートフォンでも一定期間使い続けると動作が重くなる，バッテリの持ち（以下待受時間と呼ぶ）が悪くなるなどの不満が残っており，これらはユーザビリティを著しく低下させている．

それらの不満はスマートフォンにインストールされているアプリが大きな影響を及ぼしている．しかし，高い普及率に対して，自分が使用しているスマートフォンにインストールされているどのアプリがユーザビリティに悪影響を及ぼしているかを判断できるユーザはごく少数であると考える．更にスマートフォンのホーム画面に表示されることのなく一般ユーザでは存在自体が認識されていないアプリの存在もあるため，一般ユーザが判別してアンインストールすることは困難である．また，アプリの中にはスマートフォン単体ではアンインストールが不可能なものまで存在する．そのため，一般ユーザでも簡単な操作で不要なアプリをアンインストールし最適化できるものが必要である．

現在の通信業界における大きな変化に対して，現在多くのユーザが使用しているキャリア製のスマートフォンのユーザビリティが低下している．生活の中でスマートフォンを使用する上で通信事業者と契約をして使用する場合がほとんどを占めているが，従来は大手キャリアと契約を結ぶ場合が主流であった．しかし 5 年前より格安 SIM と呼ばれている仮想移動体通信事業者（MVNO）のシェアが増加し続けており，垂直型から平行型へと大きく変化している．

MNO から MVNO へ乗り換える際にスマートフォンの買い替えが必要な場合がある．それはスマートフォンのパフォーマンスの悪化や待受時間が短いために買い換えなければならない場合である．本来ならば MNO で購入したスマートフォンでも乗り換え先の MVNO が発行する SIM カードを挿入して設定するだけで使用できる．しかし，MNO で購入したスマートフォンが乗り換えた先でも使用し続ける事ができるパフォーマンスを持っている場合に限る．そのため，通信事業者の乗り換え時においてスマートフォンの性能は重要な要素である．

また，MNO と MVNO の最も大きな違いは月額料金であり，月額料金を安く抑えるという目的で MVNO に乗り換えるのが一般的である．その目的を達成するためにも，乗り換え時に新しいスマートフォンを購入する必要の有無も重要な要素であることが分かる．

Vol.2020-GN-109 No.1
Vol.2020-CDS-27 No.1
Vol.2020-DCC-24 No.1
2020/1/23

2. 関連研究

スマートフォンのパフォーマンスが低下する問題やバッテリ消費が激しくなってしまう問題を解決するために，省電力化やレスポンス向上の手法として数多くの関連研究やアプリの存在がある．

関連研究では Android OS を改良するようなものやメーカー側のカスタマイズについて述べられているが[2] [3]，現在一般ユーザが使用しているスマートフォンの問題を解決できるようなものではない．

Android アプリとして，Play ストアにバッテリの持ちを伸ばすと謳っているものや[4][5]，不要なアプリをアンインストールする[6][7]と謳っているアプリが存在するが，それらはホーム画面からアンインストールできる一般ユーザレベルのアンインストールや Android 端末標準の設定アプリから変更できる部分を一括して操作できるに過ぎない．または, root 権限を持っていなければ対象のアプリを削除できない．また，バッテリの持ちを伸ばすと謳っているアプリは画面の輝度を低く，自動スリープ時間を短くする，アプリからの通知を無効にしたりするなどの簡易的な部分しか変更できず，バックグラウンドで動作するプリインストールアプリに関しては大半がアンインストールすることができず強制停止することに留まり，結果的に大幅な待受時間の延長やレスポンスの向上は期待できない．また，ユーザが使用しているアプリや機能を制限するということは，ユーザビリティの低下につながる．よって，バックグラウンドで動作している使用していないアプリをアンインストールするという，余計なバッテリ消耗やメモリを占有させない根本的な対策を一般ユーザのレベルで行える必要がある．

日本国内で販売されているスマートフォンの多くは大手キャリアから発売されている．そのスマートフォンにはオリジナルのスマートフォン（SIM フリー）にはインストールされていないアプリが多数含まれており，それらがスマートフォンの電池持ちや動作に大きな影響を及ぼしているという点に着目した．

本論文では，待受時間の延長とレスポンスをスムーズにし，ユーザビリティの向上を目指す．また，スマートフォンについて専門知識を有していない一般ユーザ，すなわちスマホ素人でも簡単に手持ちのスマートフォンを最適化するアプリを提案する．

改善された OS を搭載しているスマートフォンでも, インストールされているアプリが悪影響を及ぼす場合も考えられる．また，アプリストアに存在するような最適化アプリを使用しても表面上問題を解決しているように見えるが根本的な解決には至らない．そのため，スマートフォンに搭載される OS の改善と, そのスマートフォンにインストールされるアプリの取捨選択との両方の観点からのアプローチが必要である．

なお, 本論文で Android 端のみを対象としている理由として, キャリアが販売している Android 端末特有の問題であるキャリア製プリインストールアプリケーションに着目しているためである．現在日本では Android OS または iOS が搭載されているスマートフォンが半々程度に普及しているが, iOS にはキャリア製プリインストールアプリケーションは組み込まれておらず，キャリアから購入した iOS 端末を改変するといったことはない．精々キャリアのプロファイルをインストールする際にショートカットを生成する程度に留まっている．また，それらのショートカットは一般ユーザでもホーム画面から簡単にアンインストールすることができ，起動してもブラウザ上で特定のウェブサイトに接続されるといった仕組みのため，バッテリやメモリといった資源に大きな影響は及ぼしていない．

iOS 端末に対してキャリアから購入した Android 端末にはキャリアが独自に手を加えた OS が搭載されているため, 工場出荷状態からキャリア製プリインストールアプリが組み込まれている．

3. アプリの分類

スマートフォンには多くのアプリをインストールすることが可能である．それらの動作や端末に及ぼす影響はアプリによって多種多様である．また，ユーザの意思でインストールするアプリの他にも，工場出荷状態からインストールされているアプリがある．更に，ユーザの意思でインストールしたものでも実際にユーザが起動して使用しているアプリ（フォアグラウンド）の他に，起動していないのにユーザの目に見えない形で動作しているアプリ（バックグラウンド）が存在する．そのため，実際に使用しているスマートフォンにはどのようなアプリがインストールされていて，どのような動作をしているかをユーザが把握することは困難である．また，アプリによって動作やスマートフォンに及ぼす影響も様々である．

3.1 フォアグラウンドで動作するアプリ

フォアグラウンドで動作するアプリは RAM 占有率やバッテリ消耗が大きいが，ユーザの意思で起動したものであるため，端末への影響があっても許容するべきである．また, ユーザの要求によって起動されたアプリが RAM やバッテリを使用することは妥当である．

3.2 バックグラウンドで動作するアプリ

バックグラウンドで動作するアプリは，フォアグラウンドのそれよりも端末への影響は少ない．しかし，ユーザが意図してバックグラウンドで動作させているアプリより，意図せず動作してしまっているアプリの方が多い．しかし，ユーザの意図していないものでも，普段使用しているアプリの通知など必要なものもある．それらをアンインストールしてしまうと，結果的にユーザの意図とは異なる動作をしてしまう．

Vol.2020-GN-109 No.1
Vol.2020-CDS-27 No.1
Vol.2020-DCC-24 No.1
2020/1/23

3.3 ユーザが自らの意思でインストールするアプリ

図 1 はスマートフォンにインストールされているアプリの分類である．図 1 の A にあたるアプリは，ユーザの意思でインストールされたアプリの中で使用するものである．これらのアプリはその分 RAM やバッテリを多く使用するが，それはフォアグラウンドアプリと同じくリソースを有効活用しているということになる．また，ユーザがフォアグラウンドのみならずバックグラウンドでの動作を期待してインストールしたものも含まれるため，ユーザ自身が起動することはないが，アンインストールしてしまうとユーザの期待する動作ができなくなることが想定される．

図 1 の B にあたるアプリはユーザの意思でインストールされたアプリの中で使用しなくなったものや，使用するつもりでインストールしたが使わなくなったものである．これらはフォアグラウンドでは動作しないものの，バックグラウンドで動作する可能性があり，意図せず RAM やバッテリを使用されることがある．また，アプリ自体のデータ容量によってスマートフォン本体の ROM も無駄になってしまう．これらのアプリは一般ユーザがホーム画面からアンインストールすることができるほか，スマートフォンの OS によって一定期間使用されていないアプリを提示してアンインストールを促すスマートフォンも存在する．

以上のように，ユーザによってインストールされるアプリは自ら必要としてインストールする上，使用するかしないかはユーザ自身が把握している．

3.4 工場出荷状態からインストールされているアプリ

図 1 の C,D,E,F は，初期状態のスマートフォンにインストールされているアプリである．これらを本稿ではプリインストールアプリと呼ぶ．

図 1 の C,E にあたるアプリは，ユーザが使用することのあるプリインストールアプリである．これらは A や C と同じくユーザが使用するアプリであるため，バッテリやメモリを使用していたとしてもリソースを有効活用しているということになる．

図 1 の D,F にあたるアプリは，プリインストールアプリの中でユーザが使用しないアプリである．これらのアプリはユーザが不要としているアプリである上，メモリやバッテリといったリソースが使用される事がある．

よって，B と同じくアンインストールすることがリソースの節約になるが，プリインストールアプリなので一般ユーザがスマートフォンからアンインストールすることが不可能なものが多く存在する．これらを一般ユーザがアンインストールできるようにすることが必要である．

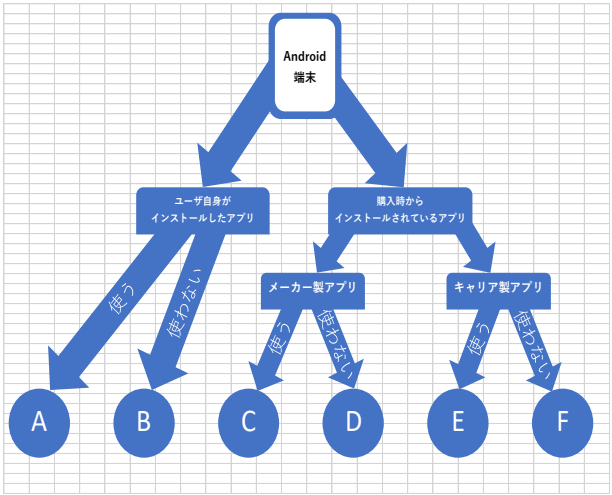

図 1 スマートフォンにインストールされているアプリケーションの分類

4. プリインストールアプリの問題点

日本で販売されている大手キャリアが販売しているスマートフォンには，数多くのプリインストールアプリが存在する．それに対して Android 端末を開発したメーカー（例えば Xperia なら SONY，Galaxy なら Samsung）が携帯通信事業者を通さず販売しているスマートフォン（SIM フリー端末）には，キャリア製プリインストールアプリはインストールされておらず，必要最低限のアプリのみがインストールされている．そのため，ユーザは自分の必要なアプリのみをインストールすることができ，限られたリソースを有効活用することができる．

しかし，各メーカーが日本で販売している機種には，SIM フリー端末としての販売を行っておらず，キャリアを通した販売のみが行われている機種が数多く存在する．そのスマートフォンを日本で購入して使用するためには，キャリアから購入する，もしくはキャリアが販売したスマートフォンを中古で入手するという手段に限られてしまう．そのようなスマートフォンにはキャリア製プリインストールアプリがインストールされているため，限られたリソースを有効活用することができない．

プリインストールアプリは日本のキャリアを通して販売されたスマートフォンに工場出荷状態からインストールされているものであり，初期化を行ってもアンインストールができない．また，プリインストールアプリの多くはスマートフォンからアンインストールできないものであるため，一般ユーザにはアンインストールする手段が存在しないと言える．

5. 基礎実験

5.1 実験内容

前節のアプリの分類により，図 1 のキャリア製プリインストールアプリをアンインストールすることで待受時間を長くすることができると予想し，基礎実験を行った．

Vol.2020-GN-109 No.1
Vol.2020-CDS-27 No.1
Vol.2020-DCC-24 No.1
2020/1/23

基礎実験では 4 台のキャリアが販売したスマートフォンを対象に実験を行った．各スマートフォンのスペックを表 1 から表 4 に示す．

表 1　実験対象端末のスペック

端末名	Galaxy S9 (SC-02K)
メーカー	Samsung
販売	docomo
OS	Android 9.0
CPU	Qualcomm Snapdragon845
RAM	4GB
ROM	32GB
バッテリ	3000mAh
ディスプレイ	1440×2960(有機 EL)

表 2 実験対象端末のスペック

端末名	Galaxy S7 Edge (SC-02H)
メーカー	Samsung
販売	Docomo
OS	Android 8.0
CPU	Qualcomm Snapdragon820
RAM	4GB
ROM	32GB
バッテリ	3600mAh
ディスプレイ	1440×2560(有機 EL)

表 3 実験対象端末のスペック

端末名	Xperia Z5（SOV32）
メーカー	Sony
販売	au
OS	Android 7.0
CPU	Qualcomm Snapdragon810
RAM	3GB
ROM	32GB
バッテリ	2900mAh
ディスプレイ	1920×1080(液晶)

表 4 実験対象端末のスペック

端末名	Xperia Z4（SOV31）
メーカー	Sony
販売	au
OS	Android 6.0
CPU	Qualcomm Snapdragon810
RAM	3GB
ROM	32GB
バッテリ	2930mAh
ディスプレイ	1920×1080(液晶)

5.2 実験環境

初期化（工場出荷状態）された 4 台の Android 端末は Play ストアへのログイン等の最低限の初期設定を行い， 4 台の端末をスリープ（無操作状態）で一定のバッテリ残量になるまで放置した．キャリア製アプリがインストールされている端末とインストールされていない端末のバッテリ残量の変動を測定し比較した．

キャリア製アプリが存在する状態のスマートフォンにはプリインストールアプリとログ取得のためのアプリのみがインストールされており，キャリア製アプリをアンインストールしたスマートフォンにはキャリア製アプリ以外のプリインストールアプリとログ取得のためのアプリのみがインストールされている．

基礎実験で使用する 4 台の Android 端末の設定は，Wi-Fi は常時接続状態，SIM カードは挿入されていないため機内モード，Bluetooth は ON など，両者同じ条件下で測定を行った．各端末の測定結果を図 2 から図 5 に示す．

なお，バッテリログは，Play ストアよりインストールし

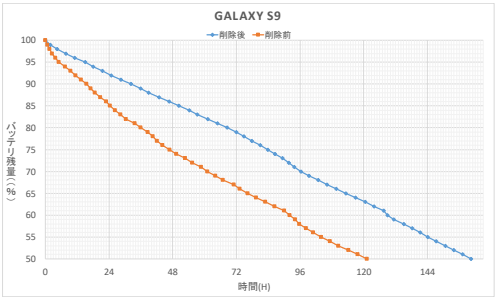

図 2 キャリア製アプリの有無による変化

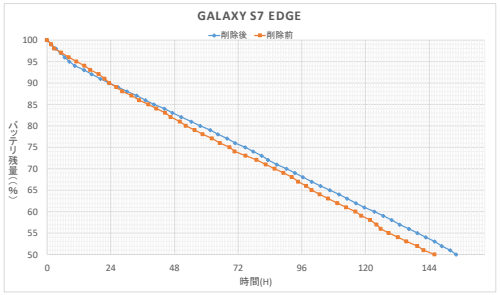

図 3 キャリア製アプリの有無による変化

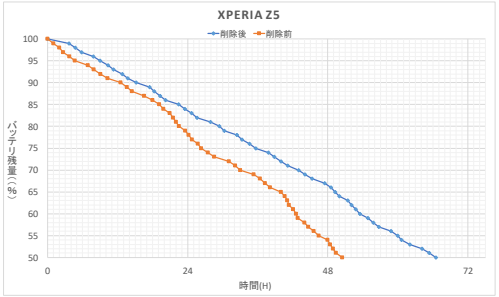

図 4 キャリア製アプリの有無による変化

Vol.2020-GN-109 No.1
Vol.2020-CDS-27 No.1
Vol.2020-DCC-24 No.1
2020/1/23

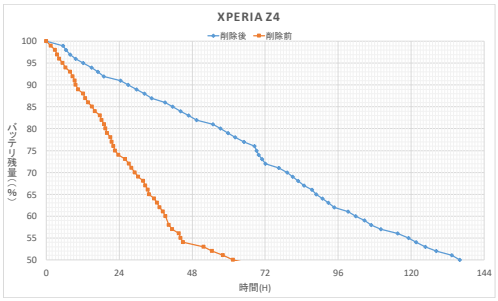

図 5 キャリア製アプリの有無による変化

たアプリ[2]により SVC 形式で取得し，グラフにした．

5.3 アプリのアンインストール方法

本論文では， Android 端末と通信が可能なコマンドラインツールである Android Debug Bridge (以下 ADB と呼ぶ)を使用する．Android 端末と PC を USB ケーブルで接続し，Package Manager(pm)ツールを使用して，インストールされているパッケージを操作することができる．ADB コマンドは Windows 上のコマンドプロンプトから実行し，接続されている Android 端末を操作できる．本論文で作成する Windows アプリケーションは，ADB コマンドを含むバッチファイルを次々実行していくことでインストールされているアプリの中からキャリア製アプリを検出，アンインストールを行う．

5.4 基礎実験結果

基礎実験の結果である図 2 から図 6 の通り，端末により変化量に差があるものの，すべてのスマートフォンにおいて無操作状態でもキャリア製プリインストールアプリが存在している状態より，存在していない状態の方が待受時間が長いことが分かった．

また，図 2 から図 5 については SIM カードを挿入していないため機内モードを ON にしていたが，図 7 では実際に契約して通信している SIM カードを挿入した状態で計測した．図 7 でもキャリア製アプリアンインストール後の方が大幅に待受時間が長くなっていることが分かった．

以上の基礎実験により，スマートフォンを無操作状態で放置した場合でもバッテリは消費され，キャリア製プリインストールアプリをアンインストールすることは待受時間の延長に有用だということが分かった．

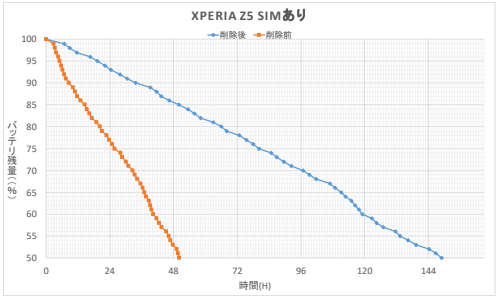

図 6 キャリア製アプリの有無による変化

6. 提案手法

6.1 キャリア販売スマートフォンの問題点

大手キャリアから販売された Android スマートフォンには，スマートフォン単体ではアンインストール不可能なアプリがプリインストールされており，それらが RAM を専有したりバックグラウンド動作がバッテリ消費による待受時間を短縮させていたりしている．そのスマートフォンから不要なアプリをアンインストールすることで，快適にそのスマートフォンを使用できるようにする．

大手キャリアで既に購入したスマートフォンを長期間使い続ける場合の問題は，キャリア製アプリをアンインストールすることで解決できることを実験で確認した．(※新規で入手した端末を使用してデータを取得中) しかし，その作業は専門知識を有する人でないと完了できず，有していたとしても数多くのアプリを個別にアンインストールするためには，手間と時間を要する．

これを解決するために，アプリを簡単にアンインストールできる Windows アプリを開発する．また，現在使用しているスマートフォンを使い続けることでスマートフォンを乗り換える必要がなくなるため，手間となるデータ移行作業が不要になる．

Android スマートフォン内にインストールされているアプリは，パッケージ名という名前で管理されている．そのパッケージ名はアプリ名とは異なり，多くはアプリ名や開発した会社の文字列から構成されている．前述したように，大手キャリア販売のスマートフォンにはキャリア独自のアプリをプリインストールしている．その点に着目し，PC に接続されたスマートフォンにインストールされているアプリから，販売元のキャリアがインストールしたアプリを特定しアンインストールすることで，待受時間やレスポンスの向上を目指す．

6.2 本アプリの動き

ADB コマンドをバッチファイルに書き込み，コマンドプロンプトでコマンドを打つこと無く一連の作業が終了できる．

6.2.1 Android 端末の接続確認

接続された Android 端末が，ADB コマンドが実行され正常に動作する状態であるかを確認するために，本アプリ起

図 7 接続要求

Vol.2020-GN-109 No.1
Vol.2020-CDS-27 No.1
Vol.2020-DCC-24 No.1
2020/1/23

動時に ADB コマンド「adb shell pm list packages」を実行する．このコマンドが正常に実行でき，返り値が 0 であれば Android 端末が正常に接続されたと判断し，本アプリのウィンドウが画面に表示される．上記の ADB コマンドを実行した際に返り値が 0 でない場合は接続が正常に行えていないと判断し，再び USB デバッグを有効化するよう促すアラートが表示される．このアラートは Android 端末が正常に接続されるまで何度も表示されるようループさせている．

ウィンドウには「解析」と「実行」の２つのボタン，その下にプログレスバーを配置する．接続が確認された段階では「解析」ボタンのみクリックできる状態とする．

6.2.2 インストールされているアプリを把握

ADB コマンド「adb shell pm list package -f >C:/hoge/list.txt」を含むバッチファイル”getlist.bat”を指定したディレクトリに生成し実行する．このコマンドでは接続された Android 端末にインストールされている全アプリのパッケージ名（ホーム画面に表示されるアプリ名とは異なり，Android 端末内部でのアプリにつけられている名前）と，そのアプリが Android 端末上の何処にインストールされているかを示すパスを”list.txt”に出力する．

6.2.3 キャリアの判別

6.2.2 で生成された”list.txt”を一行ずつ読み込み，Android 端末内にインストールされている全てのアプリのパッケージ名とディレクトリから，キャリアによって異なるパッケージ名に付けるパッケージ名やディレクトリ名のキーワード（docomo であれば「docomo,ntt,dcm,nttdocomo」，au であれば「auone,kddi」）で検索する．そのキーワードに一致する行が多かったキャリアを，接続された Android 端末の販売元として決定する．それにはコマンドライン「findstr /I “キーワード” C:/hoge/list.txt > C:/hoge/pickup.txt」を含むバッチファイル”pickup.bat”を生成，実行し”pickup.txt”に出力する．

6.2.4 キャリア製アプリを特定

6.2.3 で判別したキャリア名に関連するキーワードを引数とし，全アプリのパッケージ名とディレクトリ名からキーワードを含むパッケージ名とディレクトリ名を取得する．これには ADB コマンド「adb shell pm list package -f “キーワード" > C:/hoge/listall.txt」を含むバッチファイル”pickup.bat”を生成し，実行し”listall.txt”にディレクトリ名とパッケージ名を出力する．

6.2.5 アンインストール対象のアプリを抽出

6.2.4 で出力した”listall.txt”の文字列を使用し，アンインストール対象としたアプリの APK ファイルを，ディレクトリ”apk”に転送するコマンドライン「cd C:/hoge/apk“アンインストール対象のアプリのパス パッケージ名”」を含む文字列に変換，それを含むバッチファイル”apkpull.bat”を生成し実行する．ディレクトリ”apk”は解析開始時に作成している．これにより Android 端末からディレクトリ”apk”へアンインストール対象のアプリの APK ファイルが”パッケージ名.apk”として抽出される．

6.2.6 抽出した APK からアプリ名を特定

6.2.5 で抽出した APK ファイルをコマンドライン「aapt d badging "抽出した APK ファイルのパス" | findstr "application:" >> 抽出した APK ファイルのパス"+".txt」を含むバッチファイル”getappname.bat”を生成し実行することで，Android 端末から抽出した APK ファイルから，Android のホーム画面に表示されるようなアプリ名を含むテキストファイル”パッケージ名.txt”が生成される．このファイルは，ユーザがアンインストールするアプリをアプリ名を見て選択できる機能を今後実装するために必要となる．

6.2.7 アンインストール用 ADB コマンドに変換しバッチファイル生成

6.2.3 で生成した”pickup.txt”（キャリア製アプリのパッケージ名とパスが記述されている）の文字列を，ADB コマンド「adb shell pm uninstall -k --user 0 “パッケージ名”」に変換し，それを含むバッチファイル”del.bat”を生成する．この ADB コマンドは接続された Android 端末から”パッケージ名”に該当するアプリをアンインストールするものである．「uninstall -k --user 0」の部分は Android バージョンによって異なるため，対象となる Android 端末のバージョンを取得し正確に実行できるコマンドに変換している．

アンインストール用バッチファイル”del.bat”が生成された段階でウィンドウの左側に配置されている「解析」ボタンはクリックできなくなり，「実行」ボタンがクリックできるようになる．

6.2.8 アプリアンインストール用バッチファイル実行

6.2.7 で生成したバッチファイル”del.bat”を実行する．このバッチファイルに含まれる ADB コマンドの実行が終了した時点で２つのボタンはクリックできなくなる．

6.3 操作仕様

以下の手順でユーザは Android 端末や本アプリを操作しキャリア製アプリをアンインストールする．

1.対象となる Android 端末の設定アプリ→端末情報→ビルド番号の項目を連打し，開発者オプションを有効化する．

2.開発者オプションから開発者モードを有効化する．

3.開発者オプション内の項目から USB デバッグを有効化する．

4.本アプリが動作する PC に Android 端末を USB ケーブル(MTP)で接続する．

Vol.2020-GN-109 No.1
Vol.2020-CDS-27 No.1
Vol.2020-DCC-24 No.1
2020/1/23

5. 本アプリが上記の初期設定がれいかチッさてるをェク

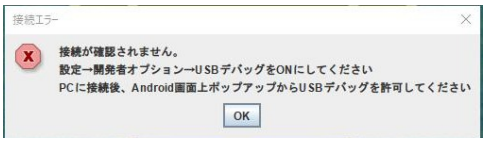

図 8 接続エラー

図 9 接続完了

図 10 解析開始

図 11 解析後

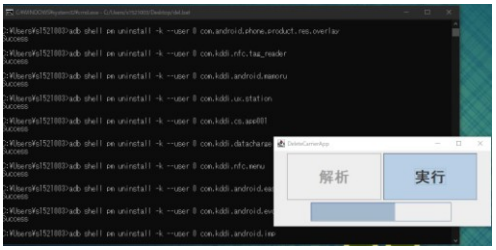

図 12 アンインストール中

図 13 アンインストール完了

(図 7 から図 9)した後，正常に接続できていたらメインウィンドウが表示される．

6.ウィンドウ左側の「解析」ボタンのみがクリックできる状態なのでクリックし，解析が終了（コマンドプロンプトのウィンドウが自動的に終了）するまで待つ．

7.解析が終了したらウィンドウ右側の「実行」ボタンがクリックできるようになるので**(図 11)**，クリックしてキャリア製アプリのアンインストールが終了するまで待つ．**(図 12)**

8.アンインストール実行が終了（コマンドプロンプトのウィンドウが自動的に終了）したら，PC と Android 端末の接続を解除する．

7. 性能評価

提案手法を用いて作成した Windows アプリ（以下，本アプリと表す）以下の項目で評価する．

表 1 から表 4 の実験端末を対象に，本アプリを使用し，キャリア製プリインストールアプリをアンインストールする．Windows 搭載のデスクトップパソコン上で本アプリを実行する．

7.1 キャリア製アプリをアンインストールする正確性

本アプリは，インストールされているパッケージ名，パス，端末の販売元という要素のみで抽出して削除している．しかし，キャリア販売の Android 端末には事前に端末側から設定の変更などを行っていなければアンインストールできないアプリが一部存在するため，本アプリが生成した削除コマンドのみではアンインストールできない状態が発生する可能性がある．それに対し，専門知識を有しているユーザがキャリア製アプリをコマンドプロンプトからアンインストールする場合は，アイコン，パッケージ名，アプリ名，通知の内容などからキャリア製アプリであるかを判別する．それらの要素から削除コマンドを実行するため，削除コマンドの正確性は高いと考えられる．

今回の実験で使用した端末にインストールされているアプリケーションの数と，本アプリを使用して検出されたキャリア製アプリの数を表５に示す．

7.2 スマートフォンの待受時間の変化

表 5 本アプリのアンインストール率

	全パッケージ数 (個)	検出数 (個)	削除数 (個)	削除率 (%)
Galaxy S9	429	60	58	96.666667
Galaxy S7 Edge	406	62	60	96.774194
Xperia Z5	425	48	48	100
Xperia Z4	419	46	46	100

Vol.2020-GN-109 No.1
Vol.2020-CDS-27 No.1
Vol.2020-DCC-24 No.1
2020/1/23

基礎実験にてキャリア製プリインストールアプリをアンインストールすることで待受時間が延びたことが分かったが，それは専門知識を有しているユーザが手動でキャリア製アプリを検出しアンインストールした際の結果である．本評価では，本アプリを用いてキャリア製アプリをアンインストールした場合，基礎実験のように一般ユーザでも行える作業のみで待受時間が延長したかを評価する．

基礎実験と同様の環境で，キャリア製プリインストールアプリがインストールされている端末のバッテリログ，専門知識を有するユーザが手動でキャリアアプリをコマンドラインからアンインストールした端末のバッテリログ，本アプリを使用してキャリア製アプリをアンインストールした端末のバッテリログを取得した．結果を図 14 から図 18 に示す．

8. 考察

8.1 キャリア製アプリをアンインストールする正確性について

表 5 の削除コマンドの結果から，異なる Android のバージョンでもアンインストールの成功率は高いと言える．しかし，２台の端末内で削除できなかった２個のアプリはいずれもキャリア製アプリであり，Android 端末を紛失した際に遠隔で初期化，ロックをかける機能を持っているものである．しかし，キャリアとの契約を解除した時点で利用不可となるアプリであることには変わらないため，本アプリの目的としてはアンインストールされるべきアプリであったといえる．それらは Android 端末側の設定アプリから「デバイス管理アプリ」という権限を外してからでないとアンインストールは不可能であった．

8.2 スマートフォンの待受時間の変化について

図 14 から図 18 の全ての端末において，キャリア製アプリがインストールされている状態に比べ，本アプリを使用しキャリア製アプリをアンインストールした状態の方が待受時間が長くなっていることが分かる．基礎実験と同じく，キャリア製アプリをアンインストールすることは待受時間が長くなった．また，図 14 と図 15 の端末では，手動でキャリア製アプリをアンインストールした状態より本アプリを使用してキャリア製アプリをアンインストールした方が待受時間が長くなっている．しかし，図 17 と図 18 ではキャリア製アプリがインストールされている状態と比べて待受時間が長くなっているものの，手動でアンインストールした状態と比べ大きく差が開いている．

9. おわりに

本論文では，移動体通信事業者が販売する Android 端末にプリインストールされているアプリが待受時間に及ぼす影響を測定した．待受時間を伸ばすためにはプリインストー

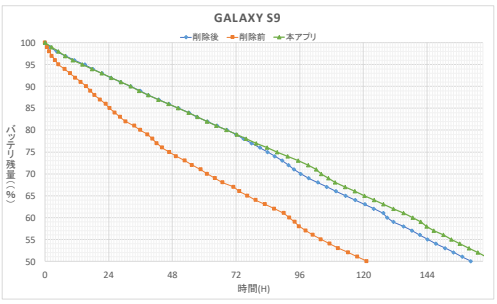

図 14 本アプリ使用時のバッテリ変動と比較

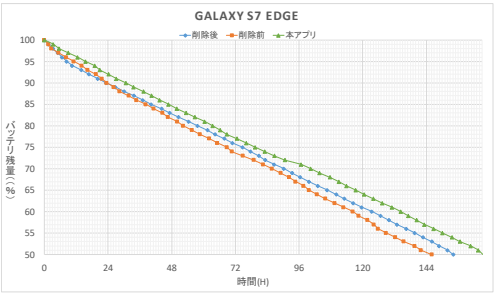

図 15 本アプリ使用時のバッテリ変動と比較

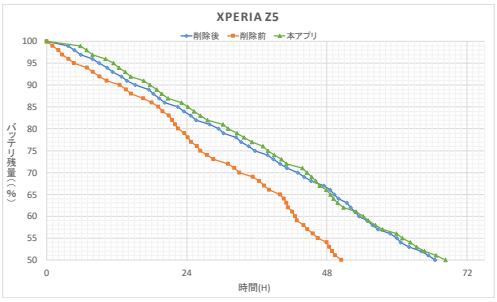

図 16 本アプリ使用時のバッテリ変動と比較

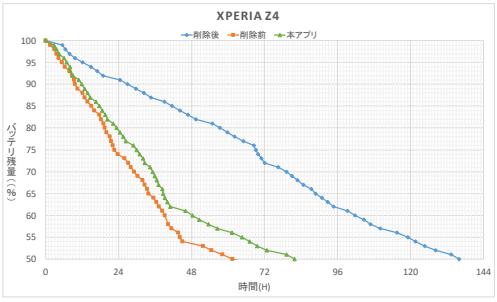

図 17 本アプリ使用時のバッテリ変動と比較

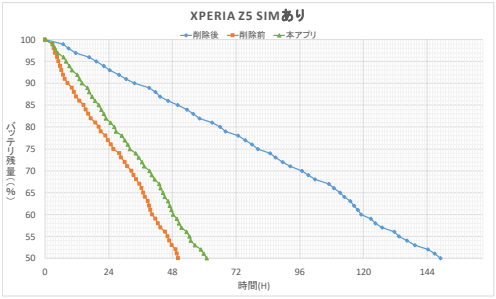

図 18 本アプリ使用時のバッテリ変動と比較

Vol.2020-GN-109 No.1
Vol.2020-CDS-27 No.1
Vol.2020-DCC-24 No.1
2020/1/23

ルアプリをアンインストールする必要があるが，アンインストールには専門知識を有していないと困難である．販売元によって異なるキャリア製アプリを一般ユーザでも簡単な操作で削除できる Windows アプリケーションを作成した．また，そのアプリを使用してキャリア製アプリを削除した結果，アンインストール前に比べて待機時間の延長に成功した．対象の端末によっては専門知識を有している人がアンインストールした場合と同等，又はそれ以上の待機時間延長に成功した．

今後はキャリア製アプリの検出の精度を向上させ更に待機時間の延長を目指す．また，本アプリを実際に一般ユーザに使用してもらい，問題なくキャリア製アプリの削除が可能か実験が必要である．また，被験者の行動や疑問をアンケートや行動観察を行い，本アプリを使いやすいものに改善していく．また，手動でキャリア製アプリをアンインストールした場合にかかる時間や手間（キーボードの打鍵回数，マウスのクリック数，不明な点を解決する情報をインターネット等で入手する作業量など）を計測し，ユーザへの負担がどのくらい低減したのかを評価していく．

また，本アプリではアプリケーションのパッケージ名，パス，端末の販売元のみから判別しているため，キャリア製プリインストールアプリケーションでも検出できていないものが複数見られた．それらを検出できるようにアプリの通信ログやパッケージの内部情報から判別できるように改良を加えることができると考える．

謝辞 MS-Word のテンプレートファイルの作成にご協力頂いた皆様に，謹んで感謝の意を表する．

10. 参考文献

[1] 総務省：“総務省｜平成 30 年版 情報通信白書｜情報通信機器の保有状況”．
http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h30/html/nd252110.html, (参照 2019-10-30).
[2] 小西 哲平，稲村 浩，川崎 仁嗣，神山 剛，大久保 信三，太田 賢：画面オフ状態におけるバックグラウンドタスク同時実行による Android 端末の省電力化，情報処理学会論文誌,55(2),587-597 (2014-02-15) , 1882-7764
[3] 野村 駿，中村 優太，坂本 寛和，濱中 真太郎，山口 実靖：Android における LRU を用いたプロセスメモリ管理，第 77 回全国大会講演論文集,2015(1),385-386 (2015-03-17)
[4] Cheetah Mobile Inc(NYSE: CMCM)：バッテリードクター（電池節約＆充電管理＆スマホ最適化），
https://play.google.com/store/apps/details?id=com.ijinshan.kbatterydoctor_en&hl=ja
[5] Color Joy：Battery Saver - バッテリードクター，
https://play.google.com/store/apps/details?id=org.artsplanet.android.sunaobattery&hl=ja
[6] INFOLIFE LLC：超便利アンインストール，
https://play.google.com/store/apps/details?id=mobi.infolife.uninstaller&hl=ja
[7] Jumobile：システムアプリ削除，
https://play.google.com/store/apps/details?id=com.jumobile.manager.systemapp&hl=ja
[8] HwangTi：Battery Log，
https://play.google.com/store/apps/details?id=kr.hwangti.batterylog&hl=ja
[9]Google Developers :“Android Debug Bridge（adb） | Android Developers”
https://developer.android.com/studio/command-line/adb.html?hl=ja, (参照 2019-12-19)