Radosław Pytlak
Politechnika Warszawska, Instytut Automatyki i Robotyki

Wojciech Stecz
Wojskowa Akademia Techniczna, Wydział Cybernetyki

Optymalizacja procesu wysyłki cukru

Optimization of sugar dispatching process

W artykule rozważany jest proces wysyłki cukru do klientów cukrowni. Podstawowym celem naszej pracy było określenie efektywności procesów logistycznych w fabryce oraz znalezienie i korekta wąskich gardeł. Zaprezentowano wybrane sposoby optymalizacji procesów logistycznych. W pracy, dla potrzeb harmonogramowania, prezentujemy zastosowanie metod heurystycznych oraz metod wykorzystujących programowanie z ograniczeniami. Praca przedstawia również dyskusję metod optymalizacji procesów logistycznych oraz omawia trudności z ich zastosowaniem.

Słowa kluczowe:
systemy gniazdowe, optymalizacja procesów logistycznych, CLP w harmonogramowaniu, programowanie z ograniczeniami.

We consider the sugar dispatching process at a sugar mill. The main goal of our work was to check an efficiency of the logistics system in the mill, find and correct the bottlenecks. Some methods for logistics processes optimization are presented. We base on the heuristics and CLP techniques for solving the scheduling problems. Some additional remarks about possibility of using the optimization methods in scheduling and logistics optimizations are presented too.

Key words:
job shop scheduling, optimization in logistics, CLP in scheduling, constraint programming.

Wprowadzenie — opis procesu wysyłki cukru

Zadanie rozwiązywane w pracy polegało na weryfikacji możliwości załadunku określonej, ustalonej *a priori* liczby ciężarówek z cukrem, z uwzględnieniem pakowania cukru w magazynie. Dla cukru pudru luzem nie występował etap pakowania, ponieważ cukier taki jest sypany do cystern, które rozwożą go do klientów cukrowni. Rozpatrywane zagadnienie wysyłki cukru wiązało się z planami zwiększenia produkcji cukru przez cukrownię. W związku z tym należało sprawdzić maksymalne możliwości wysyłkowe cukrowni oraz określić wąskie gardła systemu załadunkowego. Praca stanowi uzupełnienie analiz dotyczących procesu dystrybucji cukru (Stecz, 2009) oraz harmonogramowania przewozów, jakie można wykonywać przy pomocy o spedytorów zewnętrznych (Pytlak, Stecz, 2010). Przedstawione w pracy wyniki zostały wykorzystane praktycznie.

Proces wysyłki cukru jest procesem złożonym również ze względu na problemy związane z podprocesem pakowania cukru. Cukier wysyłany jest w torebkach o wadze 1 czy 2 kg, ale często spotyka się torebki mniejsze oraz worki, do których pakuje się większe ilości cukru. Ponadto proces pakowania i wysyłki cukru wymusza każdorazowo badanie wysyłanych partii, w szczególności tych, które nie są pakowane w torebki. Tak więc cukier pakowany przed wysyłką w worki oraz cukier zasypywany do cystern należy każdorazowo przebadać bezpośrednio przed wysłaniem, co znacząco wydłuża proces wysyłki. Biorąc pod uwagę fakt, że badaniem cukru zajmuje się zwykle laboratorium o określonej liczbie pracowników i określonej wydajności ze względu na sam proces technologiczny badania cukru,

należy zakładać konieczność równomiernego planowania załadunku ciężarówek i cystern w trakcie całej doby. Dodatkowo samochody ciężarowe nie mogą być przetrzymywane na terenie fabryki z dwóch powodów: ze względu na brak wystarczającej liczby miejsc parkingowych oraz ze względu na ograniczenia dotyczące czasu pracy kierowców, których aktywne czekanie na załadunek liczone jest jako czas efektywnej pracy, co powoduje, że muszą po kilku godzinach robić regulaminowe przerwy w pracy. Wszystkie te elementy składają się na to, że modyfikacja/optymalizacja procesu wysyłki cukru staje się zagadnieniem złożonym.

W celu weryfikacji możliwości systemu załadunkowego cukrowni należy opracować model systemu wysyłki cukru i wykonać harmonogramowanie załadunku dla zadanej liczby ciężarówek i cystern, jakie należy obsłużyć w poszczególnych dniach. Ze względu na specyfikę pracy wielu zakładów, w tym cukrowni, nie możemy pozwolić sobie na przyjęcie założenia, iż zadania niewykonane w jednym dniu można przesunąć na kolejny, ponieważ w zakładach tego typu praca trwa często przez siedem dni w tygodniu i przesunięcie zadania na kolejny dzień, przy stałym obciążeniu, powoduje znaczące opóźnienia na końcu tygodnia, co wiąże się z karami za niewypełnienie zadań zaopatrywania klientów.

Zadanie harmonogramowania wysyłki cukru

Parametry i zmienne

W opisywanym systemie uwzględniono harmonogramowanie wysyłki dla kilku produktów:
- cukru pakowanego w magazynach na palety;
- cukru pudru wysyłanego cysternami;
- produktów stanowiących dodatkowe źródło przychodów fabryki — sprzedawanych półproduktów procesu wytwórczego.

Zadanie załadunku ciężarówki lub cysterny składa się z kilku operacji, wśród których wyróżniono:
- ważenie na bramie wjazdowej i wyjazdowej (z kontrolą dokumentacji przewozowej);
- zasypywanie cysterny cukrem lub cukrem pudrem (w przypadku wysyłki cukru luzem);
- załadunek w magazynie (dla cukru wysyłanego na paletach);
- badania laboratoryjne cukru.

Każde z tych zadań można opisać czasami realizacji, które można pobrać ze zintegrowanych systemów zarządzania lub z systemów nadzorów wysyłek, które w firmach są zwykle wykorzystywane. W przypadku braku tych danych można je bardzo łatwo określić ,wykonując krótki audyt na miejscu w fabryce.

Każda operacja o identyfikatorze j przypisana jest do określonego zadania i. W dalszej części pracy zbiór numerów zadań oznaczany jest jako I, a zbiór numerów operacji należących do zadania oznaczany jest jako J^i.

Każda operacja opisywana jest (zgodnie z notacją wykorzystywaną w harmonogramowaniu zadań) kilkoma parametrami, w szczególności:
- czasem wykonania operacji j zadania i oznaczanym jako p_{ij};
- oknem czasowym dla zadania określającym przedział, w jakim można harmonogramować zadanie — $[r_{ij}, d_{ij}]$, gdzie r_{ij} to czas minimalny, od którego można planować zadanie, natomiast d_{ij} to czas maksymalny, do którego zadanie powinno być wykonane, aby uniknąć kary.

W niniejszej pracy zakładamy jedynie możliwość kar za opóźnienia i odrzucamy możliwość nieprzyjęcia wysyłki po opóźnieniu — ten przypadek raczej nie ma miejsca w procesie zaopatrywania zakładów produkcyjnych dużymi ilościami towaru. Okno czasowe pełni istotną rolę przy określaniu funkcji celu.

Dodatkowo po określeniu harmonogramu każda operacja posiada swój czas rozpoczęcia oraz czas zakończenia oznaczane odpowiednio jako: s_{ij}, e_{ij}.

Rozpatrywany w pracy przypadek cukrowni nie różni się istotnie od innych firm produkujących żywność, gdzie proces wysyłki wiąże się każdorazowo z badaniem produktu lub co najmniej sprawdzeniem dokumentacji wysyłanego towaru. W przypadku firm farmaceutycznych w grę wchodzą także inne czynności wymagane przez standardy GMP. W przypadku firm spożywczych w grę wchodzi głównie HACCP.

Funkcja celu

Ze względu na opisane powyżej uwarunkowania procesu wysyłki cukru można przyjąć, że firmy produkcyjne wysyłające towar samodzielnie do swoich klientów z magazynów lub operatorzy logistyczni obsługujący proces wysyłki towarów za producentów (mając na uwadze przepisy prawne dotyczące pracy kierowców) muszą minimalizować czas pracy w toku (ang. *Work in Progress* — WIP), czyli czas zaangażowania kierowcy w procesie załadunku. W opisanym przypadku dążymy do minimalizacji czasu załadunku, licząc od czasu wjazdu pojazdu na teren zakładu do czasu wyjazdu pojazdu z zakładu (tak bowiem, zgodnie z przepisami, należy liczyć czas pracy). Szersze omówienie kryterium czasu w toku znajduje się w dalszej części opracowania. Oznacza to, że funkcja celu ma postać:

$$F = \sum_{i \in I} \{(e_{ik} - s_{i1})\} \quad (1)$$

gdzie:
e_{ik} — czas zakończenia ostatniej operacji zadania i,
s_{i1} — czas rozpoczęcia pierwszej operacji zadania i.

Należy jednak zauważyć, że tego typu podejście ma szereg wad. W przypadku stosowania naiwnych heurystyk, szukających rozwiązania z grupy algorytmów *nondelay*, rozwiązanie mogłoby zawierać przypadki harmonogramów, które co prawda minimalizują poszczególne wartości obsługi załadunku, ale w których obsługa poszczególnych nie spełnia ograniczeń w postaci okien czasowych.

Z tego powodu funkcja celu powinna być zmodyfikowana o element związany z karą za przekroczenie zadanego okna czasowego. Postać funkcji celu uwzględniającej kary za przekroczenie okien czasowych jest następująca:

$$F = \sum_{i \in I} \{(e_{ik} - s_{i1}) + w_i \cdot \max(0, e_{ik} - d_{ik})\} \quad (2)$$

gdzie:
e_{ik} — czas zakończenia ostatniej operacji zadania i,
s_{i1} — czas rozpoczęcia pierwszej operacji zadania i,
d_{ik} — maksymalny dopuszczalny czas zakończenia ostatniej operacji zadania i.

Zadanie optymalizacji wysyłki cukru

Dla opisanego powyżej modelu wysyłki cukru z fabryki lub od operatora logistycznego do klienta sformułowanie zadania harmonogramowania opisywane jest w postaci trójki opisującej to zadanie:

$$(\alpha, \beta, \gamma) \quad (3)$$

gdzie:
α — postać systemu maszynowego (tutaj liczba punktów obsługi/gniazd wykonujących każdą z możliwych operacji),
β — parametry zadań/operacji w przypadku systemów gniazdowych,
γ — postać funkcji celu.

Omawiany w pracy system stanowi klasyczny system gniazdowy, w którym w poszczególnym gnieździe znajdują się stanowiska obsługi. Dla potrzeb niniejszej pracy za stanowisko obsługi przyjmowani są pracownicy na danym stanowisku i nie uwzględnia się dostępnego sprzętu. Uproszczenie to jest uzasadnione w sytuacji, gdy weryfikujemy możliwości wydajności systemu jako takiego i nie sprawdzamy obciążenia poszczególnych pracowników zadaniami.

Podstawowymi parametrami zadania, jakie są uwzględniane w rozwiązywaniu opisanego problemu optymalizacyjnego, są:
- okna czasowe operacji (lub okno czasowe całego zadania — w zależności od przyjętego kryterium optymalizacji) oznaczane jako $[r_{ik}, d_{ik}]$,
- zależności kolejnościowe między zadaniami i oraz k oznaczane zwykle jako $i \rightarrow k$ (co oznacza, że zadanie i jest poprzednikiem zadania k).

W przypadku nakładania kar za przekroczenie okien czasowych wprowadza się tzw. parametr wagi zadania/operacji oznaczany jako w_i lub w_{ij}.

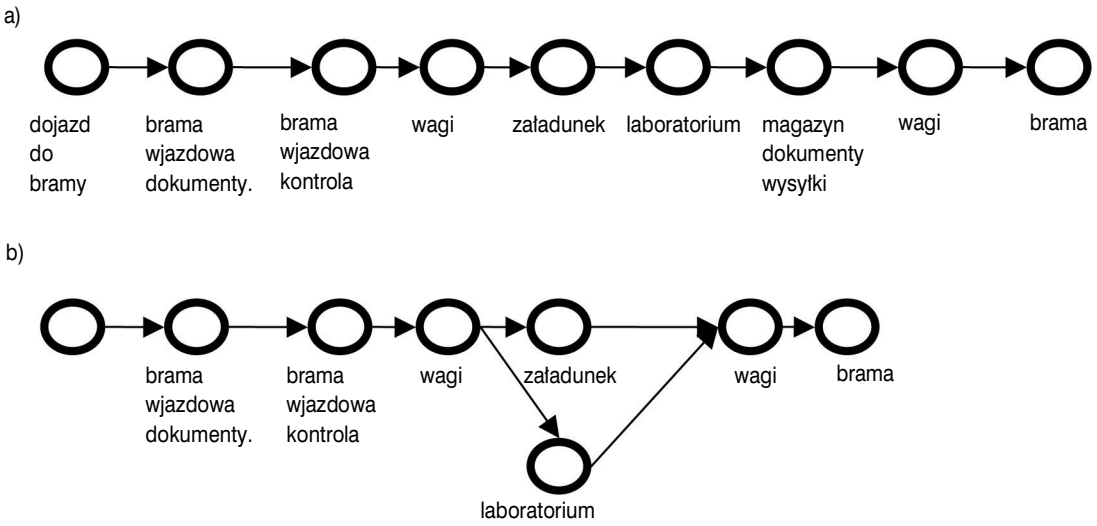
W wyniku harmonogramowania, na podstawie czasu przetwarzania operacji j należącej do zadania i (który jest ustalony z góry i oznaczany jako p_{ij}), wyznaczone zostaną wcześniej wprowadzone czasy rozpoczęcia i zakończenia operacji każdego zadania w harmonogramie oznaczane jako s_{ij} oraz e_{ij}.

W przypadku wysyłki cukru zależności kolejnościowe pomiędzy operacjami procesu wysyłki cukru na paletach i cukru luzem w cysternach pokazano na rysunku 1.

Zastosowanie metod heurystycznych do rozwiązania zadania

Najczęściej stosowanymi technikami optymalizacji w przypadku szeregowania zadań dla środowisk gniazdowych są techniki heurystyczne (Brucker, 2007; Brucker, Knust, 2006; Monch, Driefel, 2005; Neumann, Schwindt, Zimmermann, 2003; Pinedo, 2001), w tym np. algorytm przesuwania wąskich gardeł. Optymalna metoda blokowa (Grabowski, Nowicki, Smutnicki, 2003) jest niestety zbyt mało efektywna dla zadań zawierających więcej niż 10 operacji. Szczegółowe rozważania dotyczące metod harmonogramowania z wykorzystaniem technik heurystycznych wykraczają poza zakres niniejszego opracowania. Należy jednak podkreślić, że algorytmy szeregowania zadań odnoszą się do kilku podstawowych funkcji celu, wśród których jest minimalizacja pracy w toku (Liverani, 2002). Poprzez pracę w toku rozumiemy wszystkie zadania, które składają się z operacji wykonywanych w systemie gniazdowym, które w określonym momencie nie są jeszcze zakończone, ale zostały już rozpoczęte. W ten sposób WIP odnosi się głównie do stanu zużycia

Rysunek 1
Zależności kolejnościowe operacji dla zadania wysyłki cukru na ciężarówce (a) i wysyłki cukru w cysternie (b)



Źródło: opracowanie własne.

materiałów i surowców w trakcie procesu technologicznego, który nie może być uznany za zakończony. WIP uwzględnia te materiały ze stanów magazynowych, które będą zużyte (zostały zarezerwowane do zużycia) w rozpoczętym już procesie realizacji zadania.

Zastosowanie programowania z ograniczeniami do rozwiązania zadania harmonogramowania

Programowanie z ograniczeniami (ang. *Constraint Logic Programming* — CLP) stanowi alternatywny i bardzo wydajny mechanizm wykorzystywany do modelowania i rozwiązywania zadań harmonogramowania. Podstawową zaletą CLP jest deklaratywność, czyli możliwość formułowania zadania optymalizacji bez konieczności implementacji metod rozwiązania tego zadania. CLP bazuje na modelowaniu zadania jako problemu spełnienia ograniczeń CSP (ang. *Constraint Satisfaction Problem*). Ograniczenia zależą ściśle od dziedzin zmiennych, których dotyczą. Najczęściej spotykane dziedziny to dziedzina liczb całkowitych i rzeczywistych.

Główną metodą poszukiwania rozwiązania zadania jest mechanizm propagacji ograniczeń. Narzędzia implementujące metody CLP oparte są głównie na modyfikacji Prologu (CHIP, Eclipse; CISCO Systems, 2006; SICStus Prolog).

W niniejszej pracy zadania CLP rozwiązywano z wykorzystaniem mechanizmów IBM ILOG OPL (IBM ILOG, 2009), który jest językiem programowania z ograniczeniami z rozbudowanym interfejsem do modelowania na kształt systemu AMPL.

Metody wnioskowania w logice predykatów pierwszego rzędu, na jakiej oparta jest filozofia CLP, podane zostały przez T. Fruwirtha i S. Abdennadhera (Fruwirth, Abdennadher, 2003)

Poniżej zaprezentowano przykładowy program do rozwiązywania zadania harmonogramowania w systemach gniazdowych.

Pierwsza część programu zawiera deklarację zmiennych i powołanie instancji solwera CLP z biblioteki CP.

```
using CP;

int NbTasksTruck =...;
range TasksTruck = 1.. NbTasksTruck;
int NbTasksSilo =...;
range TasksSilo = 1.. NbTasksSilo;
{string} OperationNamesTruck =...;
{string} OperationNamesSilo =...;
int DurationTruck [t in OperationNamesTruck]
=...;
int DurationSilo [t in OperationNamesSilo] =...;
int Workers [t in OperationNamesTruck] =...;

tuple PrecedenceTruck {
 string pre;
 string post;
};
```

```
tuple PrecedenceSilo {
 string pre;
 string post;
};
{PrecedenceTruck} PrecedencesTruck =...;
{PrecedenceSilo} PrecedencesSilo =...;

int ReleaseDateTruck[TasksTruck] =...;
int ReleaseDateSilo[TasksSilo] =...;
```

W celu określenia harmonogramu deklarowane są zmienne modelujące przedziały czasowe dla każdego z zadań i każdej z operacji zadania. Zmienne itvsTruckWhole i itvsSiloWhole określają przedziały czasowe realizacji każdego z zadań. Zmienne itvsTruck i itvsSilo modelują przedziały czasowe/okna czasowe dla potrzeb szeregowania operacji wchodzących w skład zadań.

```
dvar interval itvsTruckWhole[h in TasksTruck]
  in 0.. maxint div 4;
dvar interval itvsTruck[h in TasksTruck][t in Operation
NamesTruck] in ReleaseDateTruck[h].. (maxint div 2) -1 size DurationTruck[t];

dvar interval itvsSiloWhole[h in TasksSilo]
 in 0.. maxint div 4;
dvar interval itvsSilo[h in TasksSilo] [t in Operation
NamesSilo] in ReleaseDateSilo[h].. (maxint div 2) –1 size DurationSilo[t];
```

Zadanie optymalizacji uwzględnia dostępność pracowników na poszczególnych stanowiskach. Oznacza to, że należy zdefiniować ograniczenia, które uniemożliwią przypisywanie operacji do stanowiska, na którym nie ma dostępnego pracownika. W tym celu definiuje się funkcje cumulFunction, które kontrolują obciążenie pracowników zadaniami. Pojęcie funkcji skumulowanej jest powszechnie znane w literaturze przedmiotu. Istnieją specjalne predykaty globalne — metody optymalizujące przetwarzanie z zastosowaniem takich funkcji.

```
cumulFunction workersUsage[t in OperationNamesTruck] =
 sum (h in TasksTruck) pulse (itvsTruck[h][t],1);
cumulFunction workersUsageSilo[t in Operation NamesSilo] =
 sum (h in TasksSilo) pulse (itvsSilo[h][t],1);
```

Poniżej zadeklarowano funkcje celu dla zadania. W przykładzie podano dwie funkcje celu, które są elementami optymalizacji wielokryterialnej. Minimalizacja następuje wg porządku leksykograficznego.

```
dexpr int goalFnt1 = max (h in TasksTruck)
 endOf (itvsTruck[h][„Gate Out”]);
dexpr int goalFnt3 = max (h in TasksSilo)
 endOf (itvsSilo[h][„Gate Out”]);

minimize staticLex(goalFnt1,goalFnt3);
```

Poniżej zdefiniowano ograniczenia zadania optymalizacji. Pokazane zostały najistotniejsze ograniczenia z punktu widzenia zadania opisywanego w pracy. Istotnym jest ograniczenie dotyczące transportu cukru w cysternach, gdzie zamiast standardowego predykatu endBeforeStart użyto predykatu startAtEnd. Zastosowanie takiego rozwiązania pomaga w zdefiniowaniu możliwości uruchomienia równoległego operacji, ale z pewnym opóźnieniem, które zdefiniowano tutaj na 10 minut. Zależności kolejnościowe operacji dla każdego z typów zadań zdefiniowano dla cukru na paletach w zmiennej PrecedencesTruck.

```
subject to {
 forall (h in TasksTruck)
   forall (p in PrecedencesTruck)
   endBeforeStart(itvsTruck[h][p.pre],
itvsTruck[h][p.post]);

 forall (h in TasksSilo)
  forall(p in PrecedencesSilo: p. post! = „Laboratory”)
    endBeforeStart(itvsSilo[h][p.pre],itvsSilo[h][p.post]);

 forall(h in TasksSilo)
  forall(p in PrecedencesSilo: p.post == „Laboratory”)
    startAtEnd (itvsSilo[h][p.post], itvsSilo[h][p.pre], 10);

 forall (i, j in OperationNamesTruck: i == j &&
 j in {„From queue”, „Gate In”, „Scales In”, „Loading”, „Laboratory”, „Scales Out”, „Gate Out”})
 workersUsage[i] + workersUsageSilo[j] <= Workers[i];

 forall (o in OperationNamesTruck: o in {„Warehouse”})
 workersUsage[o] <= Workers[o];

 forall (h in TasksTruck)
  span (itvsTruckWhole[h],
  all (t in OperationNamesTruck) itvsTruck[h][t]);

 forall (h in TasksSilo)
  span (itvsSiloWhole[h],
```

```
 all (t in OperationNamesSilo) itvsSilo[h][t]);
}
```

Ostatnie dwa ograniczenia zadania optymalizacji nie są konieczne z tego względu, że uszeregowanie operacji dla każdego z zadań wyznacza jednoznacznie przedział, w którym zadanie jest uszeregowane w harmonogramie. Niemniej jednak wprowadzenie dodatkowych przedziałów dla każdego z zadań z wykorzystaniem predykatu pokrycia span zwiększa przejrzystość zadania optymalizacji i pozwala na definiowanie dodatkowych ograniczeń na przedziałach dla zadań oraz ustalenia zależności pomiędzy nimi.

Osobną kwestią w procesie rozwiązywania zadań szeregowania są możliwości stosowania algorytmów hybrydowych (Wallace, 2005; Wallace, Schimpf, 2002), które łączą ze sobą możliwości solwerów programowania całkowitoliczbowego (Nemhauser, Wolsey, 1988; Nocedal, Wright, 1999) i solwerów CLP. W przypadku problemów harmonogramowania, w szczególności z wykorzystaniem zasobów przy realizacji określonych operacji, solwery hybrydowe spotyka się stosunkowo rzadko i tylko przy pewnych zagadnieniach dla systemów gniazdowych. Niemniej jednak ten kierunek badań jest ciągle rozwijany.

Wyniki rozwiązania zadania optymalizacji wysyłki cukru

Minimalizacja w oparciu o procedury heurystyczne

Do harmonogramowania zadań w oparciu o minimalizację pracy w toku zastosowano narzędzie Preactor APS firmy Preactor (Preactor Int., 2005). Bazą dla procedur heurystycznych wykorzystywanych w tym narzędziu jest TOC (ang. *Theory of Constraints*), która wykorzystywana jest także w procesach harmonogramowania zadań w projektach informatycznych. Możliwości narzędzia opisane zostały w pracy (Preactor Int., 2005).

Na rysunku 2 pokazano fragment harmonogramu szeregowania zadań wysyłki cukru dla ciężarówek i cystern dla kilku dni w tygodniu przy założeniu, że fabryka wysyła cukier przez 7 dni w tygodniu i przez 24 godziny na dobę.

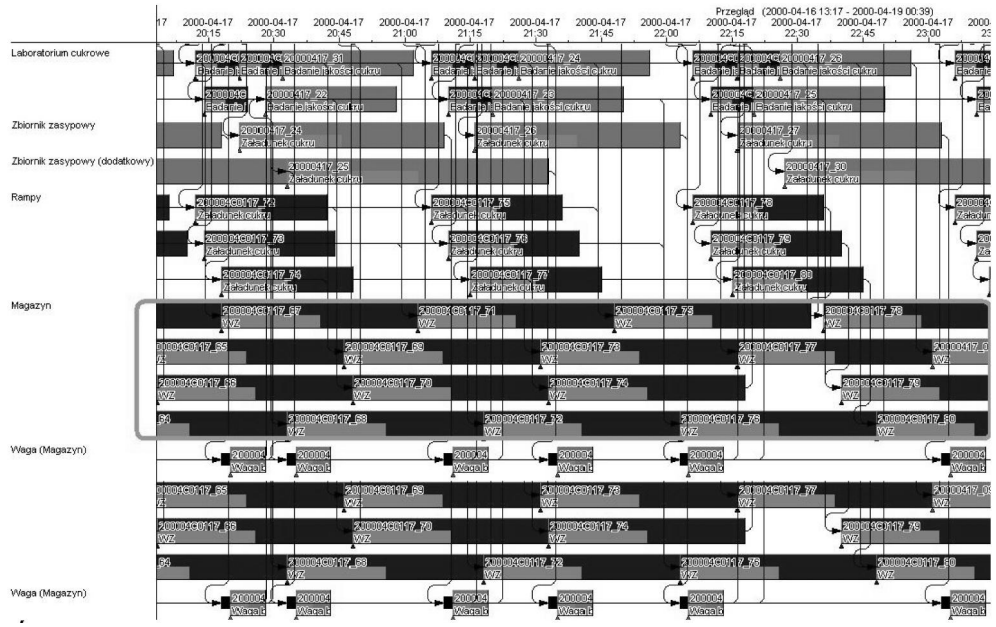
Zaznaczony prostokątem fragment pokazuje miejsce największego obciążenia zadaniami. Jest to magazyn wyrobów gotowych, gdzie oprócz pakowania w palety mają miejsce procesy przesyłania cukru do badań i przygotowywania dokumentacji wysyłkowej. Całość procesu spowalniana była jeszcze poprzez konieczność samego załadunku ciężarówek, który musiał uwzględniać kolejność układania palet (nie można bowiem układać na paletach z sa-

Rysunek 2
Wynik harmonogramowania zadań związanych z wysyłką dla kilku kolejnych dni pracy zakładu

Źródło: opracowanie własne.

Rysunek 3

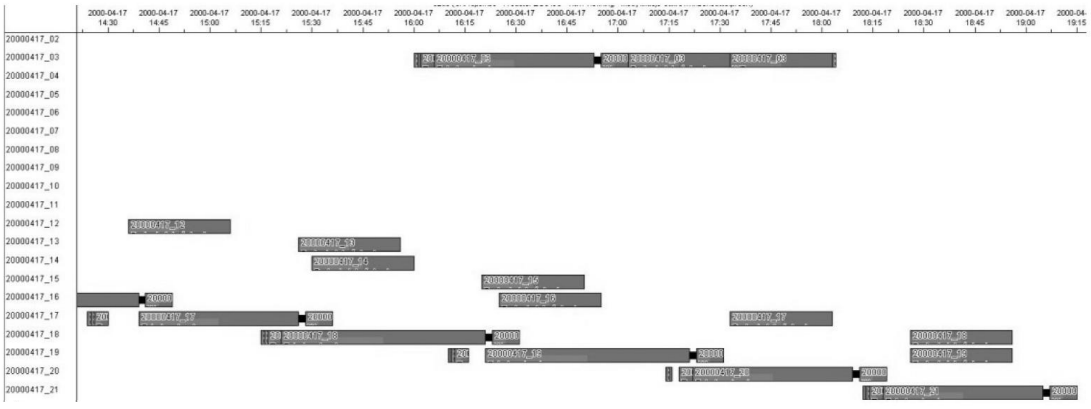
Wynik harmonogramowania zadań związanych z wysyłką dla kilku godzin w magazynie wyrobów gotowych



Źródło: opracowanie własne.

Rysunek 4

Wynik harmonogramowania dla kilku zadań dla heurystyki minimalizującej pracę w toku

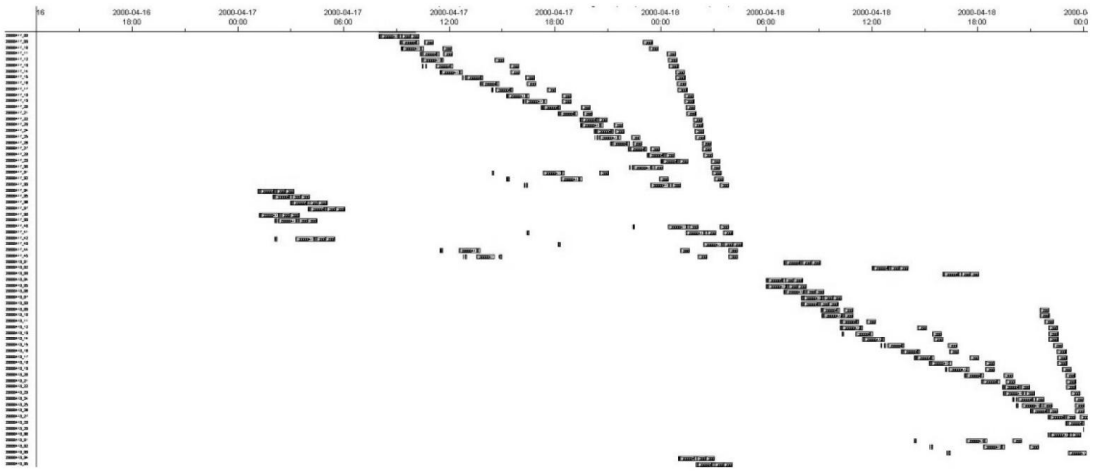


Źródło: opracowanie własne.

szetkami lub ze spakowanym w torebki cukrem pudrem).

Na rysunku 3 pokazano harmonogram wysyłki cukru dla 3 godzin danego dnia. Zaznaczono kolejność realizowanych operacji dla zadań. Widać szczegółowo dociążenie magazynów poszczególnymi operacjami. Dodatkowo widoczne jest także obciążenie wagi magazynowej, która każdorazowo, w celach kontrolnych, waży każdy pojazd po załadunku.

Rysunek 5
Efekt zastosowania algorytmu minimalizacji pracy w toku dla kilkuset zadań załadunkowych



Źródło: opracowanie własne.

Wynik działania algorytmu dla kilku wybranych zadań pokazano na rysunku 4 w postaci diagramu realizacji zadania. Na rysunku widać, że algorytm minimalizuje pracę w toku, ponieważ wszystkie operacje poszczególnych zadań są skupione w czasie.

Rysunek 5 pokazuje duży zbiór zadań w harmonogramie określonym na kilka dni. Skupienia punktów oznaczają poszczególne operacje kolejnych zadań. Widać, że nie występują zadania, których operacje byłyby rozrzucone w czasie.

Zastosowanie heurystyk pozwala na wyznaczenie harmonogramu załadunku dla kilkuset i większej liczby zadań, a jakość wyników podawanych przez algorytmy tego typu jest zadowalająca dla rozpatrywanych przypadków systemów gniazdowych. W przypadku innych systemów warto korzystać z metod optymalizacji opartych na technikach programowania z ograniczeniami.

Minimalizacja w oparciu o metody programowania z ograniczeniami

W celu przedstawienia wyników obliczeń algorytmu CLP ograniczono liczbę danych testowych — zadań do realizacji do 8, z czego 5 dotyczy zadań załadunku ciężarówek, a 3 zadań załadunku cystern. Dane testowe w formacie języka ILOG OPL pokazano poniżej.

Dane testowe:
NbTasksTruck = 5;
NbTasksSilo = 3;
OperationNamesTruck = {„From queue”, „Gate In”, „Scales In”, „Loading”, „Laboratory”, „Warehouse”, „Scales Out”, „Gate Out”};
OperationNamesSilo = {„From queue”, „Gate In”, „Scales In”, „Loading”, „Laboratory”, „Scales Out”, „Gate Out”};

Workers = [02,02,02,02,02,02,02,02];
DurationTruck = [01,01,04,30,10,45,06,01];
ReleaseDateTruck = [1, 0, 9, 12, 9];
DurationSilo = [01,01,04,120,10,06,01];
ReleaseDateSilo = [1, 0, 10];

PrecedencesTruck = {
<„From queue”, „Gate In”>,
<„Gate In”, „Scales In”>,
<„Scales In”, „Loading”>,
<„Loading”, „Laboratory”>,
<„Laboratory”, „Warehouse”>,
<„Warehouse”, „Scales Out”>,
<„Scales Out”, „Gate Out”>
};
PrecedencesSilo = {
<„From queue”, „Gate In”>,
<„Gate In”, „Scales In”>,
<„Scales In”, „Loading”>,
<„Scales In”, „Laboratory”>,
<„Loading”, „Scales Out”>,
<„Laboratory”, „Scales Out”>,
<„Scales Out”, „Gate Out”>
};

Poniżej podano wartości funkcji celu, które minimalizowano:
// solution with objective 188; 284

Przykładowy harmonogram w formacie ILOG jest następujący:

```
itvsTruckWhole =
[<1 1 143 142> <1 0 98 98> <1 9 128 119> <1 12
188 176>

<1 9 173 164>];

itvsSiloWhole =
[<1 1 253 252> <1 0 284 284> <1 10 174 164>];

itvsTruck = [
[<1 1 2 1> <1 2 3 1> <1 3 7 4> <1 7 37 30> <1
37 47 10>
 <1 91 136 45> <1 136 142 6> <1 142 143 1>]
[<1 0 1 1> <1 1 2 1> <1 2 6 4> <1 6 36 30> <1
36 46 10>
 <1 46 91 45> <1 91 97 6> <1 97 98 1>]
[<1 9 10 1> <1 10 11 1> <1 11 15 4> <1 36 66
30> <1 66 76 10>
 <1 76 121 45> <1 121 127 6> <1 127 128 1>]
[<1 12 13 1> <1 13 14 1> <1 14 18 4> <1 96 126
30>
 <1 126 136 10> <1 136 181 45> <1 181 187 6>
<1 187 188 1>]
[<1 9 10 1> <1 10 11 1> <1 18 22 4> <1 66 96
30>
 <1 96 106 10> <1 121 166 45> <1 166 172 6> <1
172 173 1>]];

itvsSilo = [
[<1 1 2 1> <1 2 3 1> <1 7 11 4> <1 126 246 120>
 <1 1 11 10> <1 246 252 6> <1 252 253 1>]
[<1 0 1 1> <1 1 2 1> <1 6 10 4> <1 157 277 120>
<1 0 10 10>
 <1 277 283 6> <1 283 284 1>]
[<1 10 11 1> <1 11 12 1> <1 16 20 4> <1 37 157
120>
 <1 10 20 10> <1 157 163 6> <1 173 174 1>]];
```

Dla zadań itvsTruckWhole <1 1 143 142> pierwsza wartość wektora oznacza, że zadanie było obowiązkowe do wykonania, wartość ostatnia oznacza czas realizacji zadania, wartość druga oznacza czas rozpoczęcia, a trzecia czas zakończenia zadania. W zmiennej itvsTruckWhole przechowywane są zagregowane czasy realizacji wszystkich operacji dla każdego z zadań, jakie zostały wprowadzone do harmonogramu. Operacjami zadania pierwszego są następujące operacje ze zmiennej itvsTruck: [<1 1 2 1> <1 2 3 1> <1 3 7 4> <1 7 37 30> <1 37 47 10> <1 91 136 45> <1 136 142 6> <1 142 143 1>]. Czas realizacji ostatniej operacji (wyjazdu z zakładu przez bramę wyjazdową) rozpoczyna się w chwili 142, trwa 1 jednostkę czasu do chwili 143.

Podsumowanie

Optymalizacja procesu wysyłki cukru miała na celu sprawdzenie przepustowości systemu logistycznego, który przygotowuje towar do wysyłki do klientów końcowych. Ze względu na dość złożony system przygotowania transportu oraz na bardzo duże ilości danych do przetworzenia nie ma możliwości zweryfikowania wydajności systemu bez zastosowania narzędzi analitycznych. Rozpatrywane w pracy zagadnienie jest szczególnie ważne w sytuacji, gdy system logistyczny musi ulec modyfikacji w celu obsłużenia większego wolumenu klientów. Zwykle bowiem wąskie gardło jest zauważane w innym miejscu niż te, które jest rzeczywistym wąskim gardłem systemu. W opisanym projekcie udało się określić najważniejsze wąskie gardła systemu, co pozwoliło na szybką modyfikację procesu wysyłki i tym samym zwiększyło przychody spółki.

Literatura

Brucker, P. (2007). *Scheduling algorithms*. Berlin: Springer.
Brucker, P., Knust, S. (2006). *Complex scheduling*. Berlin: Springer.
CISCO Systems (2006). ECLiPSe User Manual ver. 6.0.
CISCO Systems (2006). ECLiPSe Tutorial Introduction ver. 6.0.
Fruwirth, T., Abdennadher, S. (2003). *Essentials of constraint programming*. Berlin: Springer.
Grabowski, J., Nowicki, E., Smutnicki, Cz. (2003). *Metoda blokowa w zagadnieniach szeregowania zadań*. Warszawa: Akademicka Oficyna Wydawnicza EXIT.
IBM ILOG. (2009). ILOG OPL Language User's manual ver. 6.3
IBM ILOG. (2009). User's manual for CPLEX ver. 12.2
Liverani, A. (2002). *Using Genetic Algorithms to Optimise Kanban-Based Production System*. Germany: The European Applied Business Research Conference.

Monch, L., Driefel, R. (2005). *A distributed shifting bottleneck heuristic for complex job shops* (363–380). Computers and Industrial Engineering.
Nemhauser, G.L., Wolsey L.A. (1988). *Integer and Combinatorial Optimization*. New York: John Wiley & Sons.
Neumann, K., Schwindt, Ch., Zimmermann, J. (2003). *Project scheduling with time Windows and scarce resources*. Berlin: Springer.
Nocedal, J., Wright, S. J. (1999). Numerical optimization. New York: Springer-Verlag.
Pinedo, M. (2001). *Scheduling: Theory, Algorithms and Systems*. Prentice Hall.
Preactor Int. (2005). *Introduction to Scheduling*. Preactor International.
Pytlak, R., Stecz, W. (2010). *Optimizing routes for logistics operators*. Poznań: Zeszyty Naukowe Politechniki Poznańskiej.
Stecz, W. i inni. (2009). Sugar plan transportation problem reconstruction using optimization software. W: M. Fertsch, K. Grzybowska, A. Stachowiak (eds.). *Modelling of modern enterprises logistics* (25–37). Poznań: Publishing House of Poznan University of Technology.
Wallace, M. (2005). *Hybrid algorithms, local search and ECLiPSe*. CP Summer School.
Wallace, M., Schimpf J. (2002). Finding the right hybrid algorithm — A combinatorial meta-problem. *Annals of Mathematics and Artificial Intelligence,* 34, 259–269.
Wolsey, L.A., Neumhauser N.L. (1999). *Integer and combinatorial optimization*. Wiley-Interscience.