Algorytm przeszukiwania z zakazami do optymalizacji czasu komunikacji w środowisku inżynierskim CPDev

Andrzej Bożek, Dariusz Rzońca
Politechnika Rzeszowska, Katedra Informatyki i Automatyki, ul. W. Pola 2, 35-959 Rzeszów

Streszczenie: Wydajna komunikacja jest konieczna dla prawidłowej pracy rozproszonych systemów sterowania. Niniejszy artykuł poświęcony jest problemowi skrócenia czasu cyklu komunikacyjnego podczas odczytu mieszanych danych rejestrowo-binarnych w protokole Modbus ze sterownika przemysłowego, zwłaszcza implementującego pakiet inżynierski CPDev. Do optymalizacji zaprojektowano i zaimplementowano algorytm przeszukiwania z zakazami (tabu search). Wykonano obszerne eksperymenty obliczeniowe i zestawiono ich wyniki z rezultatami wcześniejszych badań, w których do tego samego problemu użyto programowania z ograniczeniami. Porównanie wyników wykazało, że nowy algorytm jest znacznie szybszy i dzięki temu nadaje się do praktycznego zastosowania w planowaniu komunikacji Modbus dla środowiska CPDev w czasie rzeczywistym.

Słowa kluczowe: tabu search, sterownik przemysłowy, PLC, komunikacja, Modbus, CPDev

1. Wprowadzenie

Protokół Modbus RTU jest powszechnie używany w systemach sterowania. Jedną z jego wad jest relatywnie niska prędkość komunikacji w porównaniu z innymi współczesnymi formami komunikacji między urządzeniami automatyki. Może to powodować ograniczenia w zastosowaniach wymagających dużej przepustowości danych w systemach sterowania w czasie rzeczywistym. W ramach niedawnych prac zaobserwowano [1, 2], że wymianę danych można znacznie przyspieszyć (rzędu kilku razy lub więcej) dzieląc odpowiednio przestrzeń adresową transmitowanych zmiennych na ramki komunikacyjne. W szczególności przygotowano algorytm optymalizacyjny oparty na programowaniu z ograniczeniami CP (ang. *constraint programming*) [1], który wyszukuje najszybszą konfigurację transmisji z uwzględnieniem dwojakiego rodzaju zmiennych (rejestrowych i binarnych) oraz z zastosowaniem dwóch funkcji protokołu (FC1 i FC3). Uzyskane wyniki są obiecujące, ale z drugiej strony badania wymagają rozszerzania z następujących przyczyn:

1. Dla istniejącego modelu CP uzyskano dowód optymalności rozwiązań tylko w przypadku zadań testowych o najmniejszych rozmiarach. Dla większości instancji testowych nie ma potwierdzenia optymalności, ani nawet miarodajnych ograniczeń dolnych czasu komunikacji.
2. Czas optymalizacji wzrasta relatywnie szybko z rozmiarem instancji testowych.
3. Użycie programowania z ograniczeniami wymaga zaawansowanego solwera, kosztownego i wykorzystującego dużo zasobów sprzętowych.

Odpowiedzią na wszystkie wymienione wątpliwości i niedogodności będzie opracowanie innego alternatywnego algorytmu do optymalizacji rejestrowo-binarnej, co jest celem niniejszej pracy. Zestawienie wyników z dwóch metod pozwoli ocenić ich spójność oraz porównać szybkość działania. Do opracowania wybrano algorytm typu przeszukiwanie z zakazami TS (ang. *tabu-search*), który będzie zaimplementowany bezpośrednio bez potrzeby użycia dodatkowego kosztownego oprogramowania i może być uruchomiony na urządzeniach z niewielkimi zasobami.

Algorytm przeszukiwania z zakazami jest metaheurystyką opartą na przeszukiwaniu lokalnym, którą po raz pierwszy zaproponował Glover [3]. Charakterystycznym elementem implementacji TS jest lista zakazów (lista tabu), która przechowuje parametry pewnej liczby ostatnio zweryfikowanych rozwiązań (tzw. atrybuty tabu) i blokuje ponowny wybór rozwiązań cechujących się tymi parametrami, aby uniemożliwić zapętlenie przeszukiwań w zamkniętej trajektorii. Zastosowanie metaheurystyki TS dla konkretnego problemu optymalizacji wymaga: (a) zaprojektowania sposobu kodowania rozwiązań za pomocą zmiennych decyzyjnych, (b) określenia dozwolonych zmian zmiennych decyzyjnych, zwanych ruchami, przy tworzeniu nowego rozwiązania na podstawie poprzedniego, (c) specyfikacji sąsiedztwa, tzn. zbioru wszystkich możliwych ruchów wykonywanych na jednym rozwiązaniu, (d) zdefiniowania atrybutu tabu, (e) określenia innych szczegółowych założeń i parametrów.

Autor korespondujący:
Dariusz Rzońca, drzonca@kia.prz.edu.pl

Artykuł recenzowany
nadesłany 15.12.2024 r., przyjęty do druku 16.07.2025 r.

2. Przegląd literatury

Norma IEC 61131-3 [4] wdrożona w Polsce jako PN-EN 61131-3 definiuje następujące dziedzinowe języki programowania dla sterowników przemysłowych PLC (ang. *Programmable Logic Controller*) i PAC (ang. *Programmable Automation Controller*): ST (ang. *Structured Text*), IL (ang. *Instruction List*), FBD (ang. *Function Block Diagram*), LD (ang. *Ladder Diagram*), SFC (ang. *Sequential Function Chart*). Jednym z pakietów inżynierskich implementujących języki normy IEC 61131-3 jest CPDev (ang. *Control Program Developer*) [5, 6] opracowany w Katedrze Informatyki i Automatyki Politechniki Rzeszowskiej. Środowisko to, mimo akademickiego a nie komercyjnego rodowodu, zostało wdrożone w kilku rozwiązaniach przemysłowych, we współpracy z przedsiębiorstwami polskimi i zagranicznymi (np. Praxis Automation Technology B.V. z Holandii czy iGrid T&D z Hiszpanii). Budowa CPDev pozwala na wykorzystanie różnych platform sprzętowych, od niewielkich ośmiobitowych procesorów do architektury wielordzeniowej [7] czy FPGA [8], dzięki użyciu dedykowanych interpreterów zwanych maszynami wirtualnymi CPDev [9]. Program sterowania, przygotowany w środowisku CPDev na komputerze PC w jednym z języków normy IEC 61131-3, jest kompilowany do kodu pośredniego VMASM (ang. *Virtual Machine Assembler*) a następnie interpretowany na docelowej platformie sprzętowej przez maszynę wirtualną CPDev [10]. Jedną z charakterystycznych cech takiego rozwiązania jest jednakowe rozmieszczenie zmiennych programu sterowania w pamięci danych maszyny wirtualnej, niezależne od architektury fizycznej użytego mikrokontrolera, co umożliwia przedstawioną w tym artykule optymalizację transmisji.

Zagadnienia związane z wymianą danych w rozproszonych systemach automatyki są szeroko omawiane w literaturze. Zbiorczy przegląd można znaleźć w monografiach [11, 12]. Norma IEC 61158 [13] przyjęta w Polsce jako PN-EN 61158 definiuje sieci polowe stosowane do komunikacji w przemysłowych systemach automatyki. Współcześnie często stosowane są także rozwiązania bazujące na przemysłowych sieciach Ethernet [14] oraz architektury heterogeniczne, integrujące różne sieci [15]. W niewielkich systemach wciąż jednak wykorzystywana jest komunikacja poprzez port szeregowy (np. RS-485), za pomocą przemysłowych protokołów komunikacyjnych. Wśród nich popularny jest protokół Modbus.

Modbus jest protokołem typu *master-slave*, co oznacza, że urządzenie nadrzędne inicjuje wymianę danych, a urządzenie podrzędne odpowiada. W jednej ramce komunikacyjnej można przesłać wiele zmiennych (zależnie od ich typu), dzięki czemu odpowiednie grupowanie danych może prowadzić do skrócenia łącznego czasu cyklu komunikacyjnego. Takie podejście było niekiedy przedstawiane w literaturze. W artykule [16] rozważana jest problematyka grupowania danych w protokole Modbus TCP podczas komunikacji z panelem HMI. W pracach innych autorów opisywano wpływ rozmieszczenia zmiennych na wydajność transmisji w protokole Modbus [17, 18], proponowano rozszerzenia tego protokołu [19] prowadzące do poprawy parametrów czasowych i analizowano wyniki eksperymentalne [20]. Innym aspektem rozważanym w [21] było dynamiczne zastosowanie grupowania podczas przeciążenia magistrali komunikacyjnej.

Użyty w tej pracy algorytm przeszukiwania z zakazami od czasu swojego wprowadzenia [3] zdobył dużą popularność i okazał się bardzo skuteczny w optymalizacji dyskretnej. Jest szeroko stosowany do chwili obecnej w różnorodnych zadaniach optymalizacji. W najnowszych pracach TS zastosowano m.in. do harmonogramowania zadań w problemie przepływowym z maszynami równoległymi [22], optymalizacji energii do podgrzewania wody [23], optymalizacji struktury systemów dystrybucji energii elektrycznej [24], czy też minimalizacji liczby przecięć krawędzi w rysowaniu grafów [25].

Bieżący artykuł jest kontynuacją poprzednich prac: [1, 2, 26, 27]. Artykuły [26, 27] koncentrują się na grupowaniu podczas wymiany danych między sterownikiem PLC a panelem HMI wyłącznie za pomocą funkcji rejestrowych Modbus (FC3, FC6, FC16). Funkcje bitowe (jak np. FC1) nie są tam rozważane. W pracy [2] problematykę grupowania przedstawiono w sposób szerszy, dla różnych protokołów komunikacyjnych, ale nadal zakładając wyłącznie transmisję rejestrową, a nie binarną. Podobne założenia jak w niniejszym artykule (grupowanie przy mieszanym odczycie rejestrowo-binarnym w protokole Modbus) przyjęto w [1]. Opisano tam poprzednio uzyskane wyniki optymalizacji dla programowania z ograniczeniami. W bieżącym artykule użyto innego algorytmu optymalizacyjnego, przeszukiwania z zakazami i porównano oba podejścia.

3. Grupowanie danych w protokole Modbus

Protokół Modbus obsługuje dwa typy zmiennych – binarne i rejestrowe (szesnastobitowe). W sterownikach Modicon były one umieszczone w czterech osobnych przestrzeniach adresowych, odpowiednio jako wyjścia – zmienne binarne (ang. *Output Coils*), wejścia binarne (ang. *Discrete Inputs*), rejestry wejściowe (ang. *Input Registers*) i szesnastobitowe zmienne – rejestry (ang. *Holding Registers*). Wejścia (binarne i rejestrowe) są tylko do odczytu, pozostałe zmienne można odczytywać i zapisywać. Do ich obsługi w protokole Modbus zaimplementowano następujące funkcje: FC1 – odczyt zmiennych binarnych, FC2 – odczyt wejść binarnych, FC3 – odczyt rejestrów, FC4 – odczyt rejestrów wejściowych, FC5 – zapis pojedynczej zmiennej binarnej, FC6 – zapis pojedynczego rejestru, FC15 – zapis wielu zmiennych binarnych, FC16 – zapis wielu rejestrów.

We współczesnych sterownikach przemysłowych programowanych w językach normy IEC 61131-3 obsługiwane są także inne typy zmiennych, nie tylko binarne czy szesnastobitowe. Norma definiuje kilkanaście różnych typów, a także pozwala na tworzenie zmiennych złożonych – tablic i struktur. Typy te nie są natywnie obsługiwane w Modbus. Podczas transmisji wymagana jest konwersja do obsługiwanego w protokole typu, albo transmitowaną grupę rejestrów traktuje się jako fragment pamięci sterownika, zawierający kilka zmiennych, a ich wyodrębnienie i interpretacja odbywa się po stronie odbiorcy.

Podział przestrzeni adresowej jest więc często inny niż założony w Modbus. W środowisku inżynierskim CPDev wszystkie zmienne, niezależnie od typu, umieszczone są we wspólnej przestrzeni adresowej. Zmienna binarna typu BOOL (zajmująca w pamięci sterownika implementującego CPDev jeden bajt) może więc bezpośrednio sąsiadować ze zmienną szesnastobitową (dwubajtową) typu INT czy WORD. Podejście takie ułatwia obsługę struktur, które mogą zawierać sąsiadujące zmienne różnych typów. Warto jednak zauważyć, że na niektórych architekturach mikroprocesorowych (np. ARM) dostęp do zmiennych ułożonych bez wyrównania (np. zmienna trzydziestodwubitowa pod adresem niepodzielnym przez cztery) musi odbywać się w odpowiedni sposób [28].

Jak wspomniano grupę rejestrów transmitowaną w protokole Modbus można traktować jako fragment pamięci sterownika, zawierający kilka zmiennych różnych typów. Skoro w przestrzeni adresowej sterownika CPDev zmienne binarne sąsiadują ze zmiennymi szesnastobitowymi (rejestrami), to decyzją projektanta systemu jest czy użyć funkcji FC3 do odczytu wszystkich zmiennych (także binarnych), czy też zmienne binarne odczytywać osobno dedykowaną funkcją FC1. Oczywiście w takim przypadku należy pamiętać o odpowiednim przeliczeniu adresów. Adres binarny, wykorzystywany w funkcji FC1, będzie zwiększany o jeden wraz z każdym bajtem,

a adres rejestrowy (wykorzystywany w funkcji FC3) będzie zwiększany o jeden dla każdego szesnastobitowego rejestru (czyli co dwa bajty). Dodatkowo kilka niesąsiadujących ze sobą zmiennych (zarówno rejestrowych jak też binarnych) można odczytać w kilku osobnych komunikatach, lub też łącznie, jako jedną grupę (blok) kilkunastu rejestrów, przesyłając także niewymagane dane. Możliwe jest także, że niektóre zmienne binarne będą przesyłane dwukrotnie zarówno w pewnym bloku rejestrowym funkcją FC3, jak też w bloku binarnym funkcją FC1. Mimo nadmiarowości takie postępowanie może niekiedy być korzystne, gdyż unika się dzielenia bloków na części, co zwiększa liczbę transmitowanych ramek i wydłuża łączny czas cyklu komunikacyjnego. Wybór odpowiedniego schematu wymian danych rzutuje na uzyskiwane parametry czasowe. Jak pokazano w [1] wybór optymalnego grupowania jest złożonym zagadnieniem, przy większej liczbie zmiennych celowe jest użycie algorytmów optymalizacyjnych. Znalezienie najlepszego grupowania podczas przygotowywania programu sterowania na komputerze PC pozwala wyznaczyć odpowiedni scenariusz wymian danych, który następnie będzie cyklicznie realizowany na sterowniku PLC.

4. Algorytm przeszukiwania z zakazami

4.1. Kodowanie rozwiązania

Szczegółowa sformalizowana definicja problemu optymalizacji rejestrowo-binarnej została przedstawiona w pracy [1]. Zestaw danych opisujących instancję tego problemu ma postać krotki

$$\mathcal{P} = \left(R, L, H, \alpha_{\mathrm{R}}, \beta_{\mathrm{R}}, \omega_{\mathrm{R}}, \alpha_{\mathrm{B}}, \beta_{\mathrm{B}}, \omega_{\mathrm{B}}\right), \tag{1}$$

gdzie R jest zbiorem adresów zmiennych rejestrowych wymagających odczytu, L i H są zbiorami adresów zmiennych binarnych (umieszczonych odpowiednio w dolnej lub górnej połowie lokalizacji rejestrowych), parametry α_{R}, β_{R} są współczynnikami funkcji liniowej określającej czas odczytu bloku zmiennych rejestrowych na podstawie jego długości, ω_{R} reprezentuje limit długości bloku zmiennych rejestrowych, dla bloków zmiennych binarnych parametry α_{B}, β_{B}, ω_{B} mają analogiczne znaczenie.

Wprowadzimy pomocnicze definicje

$$V = R \cup L \cup H, \quad (v_k)_{k=1}^{|V|} \subseteq V, \quad v_k < v_{k+1}, \tag{2}$$

$$B = 2L \cup 2H + 1, \quad (b_k)_{k=1}^{|B|} \subseteq B, \quad b_k < b_{k+1}, \tag{3}$$

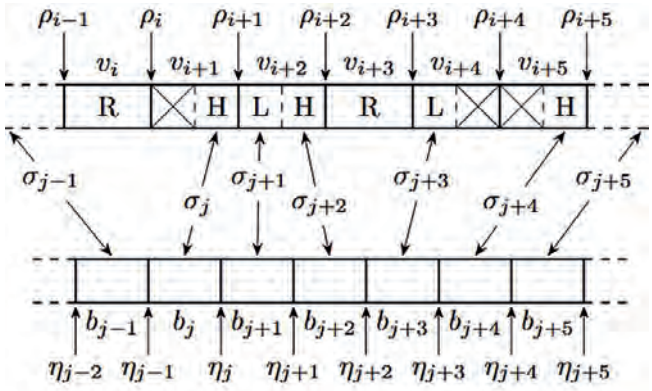
określające ciągi zawierające rosnąco uporządkowane adresy wszystkich odczytywanych lokalizacji rejestrowych (v_k), niezależnie od rodzaju danych R/L/H, oraz kolejno numerowanych (z rozdzielczością ośmiobitową) lokalizacji zajętych tylko przez zmienne binarne (b_k). Adekwatnie do zasad adresowania zmiennych i zgodnie z (3), lokalizacji binarnej b_k odpowiada adres rejestrowy $\lfloor b_k / 2 \rfloor$.

Do zakodowania rozwiązania w prezentowanym algorytmie przeszukiwania z zakazami wykorzystano strukturę danych złożoną z trzech ciągów binarnych zmiennych decyzyjnych

- $(\sigma_k)_{k=1}^{|B|} \in \{0,1\}$ – lokalizacja binarna b_k musi zostać przesłana w bloku binarnym, gdy $\sigma_k = 1$, i musi zostać przesłana w bloku rejestrowym w przeciwnym przypadku,
- $(\rho_k)_{k=1}^{|V|-1} \in \{0,1\}$ – lokalizacje rejestrowe v_k i v_{k+m} są transmitowane w odrębnych blokach wtedy i tylko wtedy, gdy $\rho_k = 1$,
- $(\eta_k)_{k=1}^{|B|-1} \in \{0,1\}$ – lokalizacje binarne b_k i b_{k+m} są przesyłane w odrębnych blokach wtedy i tylko wtedy, gdy $\eta_k = 1$,

gdzie v_{k+m}, $m \geq 1$ oznacza kolejną lokalizację rejestrową transmitowaną po v_k, analogicznie jest dla lokalizacji binarnych b_k i b_{k+m}.

Na rys. 1 przedstawiony został przykład pozwalający dokładniej wyjaśnić zasady kodowania. W górnej części znajduje się fragment ciągu (v_k), tj. parametry v_i, ..., v_{i+5}, które zawierają adresy lokalizacji rejestrowych w rozumieniu (2), tzn. uporządkowanych lokalizacji, które muszą (R) lub mogą (L i H) być przesyłane w blokach rejestrowych. Analogicznie w dolnej części umieszczony jest fragment ciągu b_k w postaci parametrów b_{j-1}, ..., b_{j+5}, które zawierają rosnąco uporządkowane adresy lokalizacji binarnych w rozumieniu (3), możliwe do odczytu w blokach binarnych. Zmienne $\sigma_{j-1},\ldots,\sigma_{j+5}$, gdy mają wartość 1, wymuszają transmisję lokalizacji b_k o tym samym indeksie w bloku binarnym, a w przeciwnym razie w bloku rejestrowym w lokalizacji jednoznacznie określonej relacją $v_i = \lfloor b_j / 2 \rfloor$.



Rys. 1. Kodowanie problemu za pomocą zmiennych decyzyjnych
Fig. 1. Problem encoding by decision variables

Zasadniczo zmienna ρ_k decyduje o transmisji lokalizacji v_k i v_{k+1} w jednym bloku (= 0) lub ich rozdzieleniu (= 1), ale jest tak tylko wtedy, gdy obie te lokalizacje są przydzielone do bloków rejestrowych. Jeśli lokalizacja v_k nie jest przydzielona do bloku rejestrowego (tzn. zawiera zmienne binarne przyporządkowane przez (σ_j) wyłącznie do bloków binarnych), wtedy wartość ρ_k jest ignorowana w dekodowaniu rozwiązania i określana jako *redundantna*. Jeśli natomiast v_k podlega transmisji rejestrowej, ale v_{k+1} nie, wtedy ρ_k determinuje separację bloków między v_k, a najbliższą lokalizacją v_{k+m} ujętą w transmisji (o ile taka istnieje). Powracając do przykładu z rys. 1, jeśli $\sigma_j = 0$, to ρ_i będzie określać separację transmisji między v_k i v_{k+1}, natomiast w przypadku $(\sigma_j, \sigma_{j+1}, \sigma_{j+2}) = (1,1,0)$ określi separację między v_k i v_{k+2}, a zmienna ρ_{i+1} stanie się redundantna. Gdyby jednak otrzymać $(\sigma_j,\ldots,\sigma_{j+4}) = (1,\ldots,1)$, wtedy ρ_i będzie określać separację między v_k i v_{k+3}, w czym wartości σ_{j+3} i σ_{j+4} nie mają znaczenia, ponieważ v_{k+3} musi być przesłane w bloku rejestrowym. Analogicznie zmienna decyzyjna η_k jest uznawana za redundantną, jeśli $\sigma_k = 0$ wyłącza b_k z transmisji w bloku binarnym. Na przykład, jeśli $(\sigma_j,\ldots,\sigma_{j+3}) = (1,0,0,1)$ (rys. 1), wtedy η_j specyfikuje separację bloków binarnych między b_j i b_{j+3}, a zmienne η_{j+1} i η_{j+2} stają się redundantne.

W oparciu o omówioną wyżej interpretację ciągów zmiennych decyzyjnych (σ_k), (ρ_k) i (η_k) można jednoznacznie podzielić odczytywane zmienne na bloki rejestrowe i binarne, uzyskując zbiory oznaczone w pracy [1] jako F_{B} (2) i F_{R} (3), następnie formuła (4) z tej samej pracy pozwala wyznaczyć czas cyklu komunikacyjnego, będący dla rozważanego tu algorytmu minimalizowaną funkcją celu.

4.2. Ruchy i sąsiedztwo

W algorytmach optymalizacji bazujących na przeszukiwaniu lokalnym sąsiedztwo jest zbiorem rozwiązań, które można utworzyć przez modyfikacje rozwiązania bieżącego, zwane ruchami.

Zdefiniujmy ciąg $(\gamma_i)_{i=1}^{2|B|+|V|-2} = \left((\sigma_k)(\rho_k)(\eta_k)\right)$ stanowiący konkatenację trzech ciągów zmiennych decyzyjnych oraz łatwą do praktycznej implementacji funkcję $\mathcal{R}\left((\gamma_i), i\right) \in \{T, F\}$, która zwraca wartość logiczną T (true) wtedy i tylko wtedy, gdy zmienna zawarta w ciągu (γ_i) pod indeksem i jest redundantna w sensie zdefiniowanym w poprzedniej sekcji. Ciąg (γ_i) będzie nazywany *rozwiązaniem*, ponieważ koduje jednoznaczne rozwiązanie problemu.

W zaimplementowanym algorytmie za sąsiedztwo rozwiązania (γ_i) przyjęto zbiór

$$\mathcal{N}\left((\gamma_i)\right) = \bigcup_{\substack{1\le j\le M \\ \mathcal{R}((\gamma_i),j)=F}} \left\{(\gamma_i^*)_{i=1}^{M} \mid \gamma_i^* = (1-\delta_{ij})\gamma_i + \delta_{ij}(1-\gamma_i)\right\}, \quad (4)$$

gdzie $M = 2|B| + |V| - 2$, a δ_{ij} jest symbolem Kroneckera. Ruchy są zatem uzyskiwane przez negację logiczną pojedynczych zmiennych decyzyjnych z rozwiązania wyjściowego, z wyłączeniem zmiennych redundantnych, a sąsiedztwo jest zbiorem tych ruchów. Modyfikacje zmiennych redundantnych nie powodują zmiany zakodowanego rozwiązania, więc ich wykluczenie jest oczywistym założeniem przy tworzeniu sąsiedztwa umożliwiającego wydajne przeszukiwanie.

4.3. Atrybut tabu

Atrybut tabu to cecha rozwiązania umieszczana na liście zakazów. Jeśli atrybut tabu nowego rozwiązania w przeszukiwanym sąsiedztwie znajduje się na tej liście, to rozwiązanie takie jest pomijane. Jest to podstawowy mechanizm algorytmu TS zabezpieczający przed utknięciem przeszukiwania w cyklicznej trajektorii.

Dla rozważanego problemu optymalizacji przetestowano klika konstrukcji atrybutu zakazu. Pierwsza jako atrybut przyjmuje indeks zmiennej decyzyjnej w ciągu (γ_i), która została zmodyfikowana w ruchu generującym nowe rozwiązanie. Taki atrybut tabu okazał się jednak zbyt mało specyficzny dla konkretnego rozwiązania, ponieważ pomija wszystkie wartości z (γ_i), poza jedną, i w efekcie słabo zabezpiecza przed trajektoriami cyklicznymi. Sprawdzono wobec tego konstrukcję o przeciwnym charakterze, traktującą cały ciąg (γ_i) jako atrybut zakazu maksymalnie specyficzny dla reprezentowanego rozwiązania. Ten wybór okazał się znacznie lepszy, natomiast dodatkowe badania eksperymentalne wykazały, że algorytm jeszcze efektywniej minimalizuje funkcję celu po doprecyzowaniu atrybutu zakazu dla rozwiązania (γ_i) do postaci

$$\mathcal{T}\left((\gamma_i)\right) = (\tau_k)_{k=1}^{|B|+|V|-2},$$
$$\tau_k = \begin{cases} \gamma_{k-|B|}, & \mathcal{R}\left((\gamma_i), k-|B|\right) = F \\ 0, & \mathcal{R}\left((\gamma_i), k-|B|\right) = T \end{cases}, \quad (5)$$

tzn. atrybut zakazu jest konkatenacją ciągów zmiennych decyzyjnych (ρ_k) i (η_k) z pominięciem (σ_k), dostrajającym specyficzność atrybutu, i z zamaskowaniem zmiennych redundantnych wartościami 0.

4.4. Inne elementy algorytmu

Oprócz wyżej wymienionych założeń projektowych i funkcji, opracowany algorytm TS uwzględnia jeszcze inne, z których najważniejszymi są:

1. **Wykrywanie rozwiązań niedopuszczalnych.** Jeśli ciąg binarny (γ_i) reprezentuje rozwiązanie z blokiem zmiennych rejestrowych dłuższym niż ω_R lub blokiem zmiennych binarnych dłuższym niż ω_B, wtedy rozwiązanie jest uznawane za niedopuszczalne i pomijane w przeszukiwanym sąsiedztwie.
2. **Generowanie rozwiązania początkowego.** Rozwiązanie początkowe tworzone jest jako losowy ciąg (γ_i). W razie uzyskania rozwiązania niedopuszczalnego, losowanie jest powtarzane.
3. **Kryterium aspiracji.** Zastosowano standardowe dla TS kryterium aspiracji, polegające na akceptacji nowego rozwiązania poprawiającego dotychczasowe optimum globalne, nawet jeśli jest ono wykluczone przez listę tabu.
4. **Stochastyczne zawężenie sąsiedztwa.** Zamiast bezpośredniego zastosowania pełnego sąsiedztwa $\mathcal{N}\left((\gamma_i)\right)$ zdefiniowanego w (4), każdy jego element jest uwzględniany z prawdopodobieństwem p_N, co pozwala ograniczyć rozmiar sąsiedztwa i zdywersyfikować je stochastycznie.
5. **Perturbacja rozwiązań.** Dla lepszego zdynamizowania przeszukiwań i rozbijania ewentualnych trajektorii cyklicznych, rozwiązanie jest zaburzane co każde D_P iteracji poprzez zanegowanie logiczne D_N losowo wybranych zmiennych w ciągu (γ_i).

5. Eksperymenty obliczeniowe

Do eksperymentów wykorzystano instancje testowe opracowane dla wcześniejszej weryfikacji optymalizacji CP w pracy [1], aby porównać wyniki poprzednio użytej metody z nowym algorytmem TS. Do obliczeń użyto 50 rdzeni serwera obliczeniowego z procesorem AMD EPYC Rome i 120 GB pamięci RAM. Zestaw testowy zawiera po 10 instancji o rozmiarach 20, 60 i 180 zmiennych do odczytu. W celu porównawczym optymalizację przeprowadzono również dla tych samych trzech prędkości transmisji: 9600, 38 400 i 115 200 b/s. Dla najmniejszej liczby zmiennych, tj. 20, algorytm TS natychmiast uzyskał wyniki identyczne z CP dla wszystkich trzech prędkości transmisji i 10 instancji testowych. Przyjęto wobec tego, że są to wyniki optymalne (dla większości z nich również solwer CP potwierdził optymalność, patrz [1]) i wykluczono je z dalszej analizy. Szczegółową analizę statystyczną wyników przeprowadzono dla pozostałych instancji testowych, tj. zawierających 60 i 180 zmiennych.

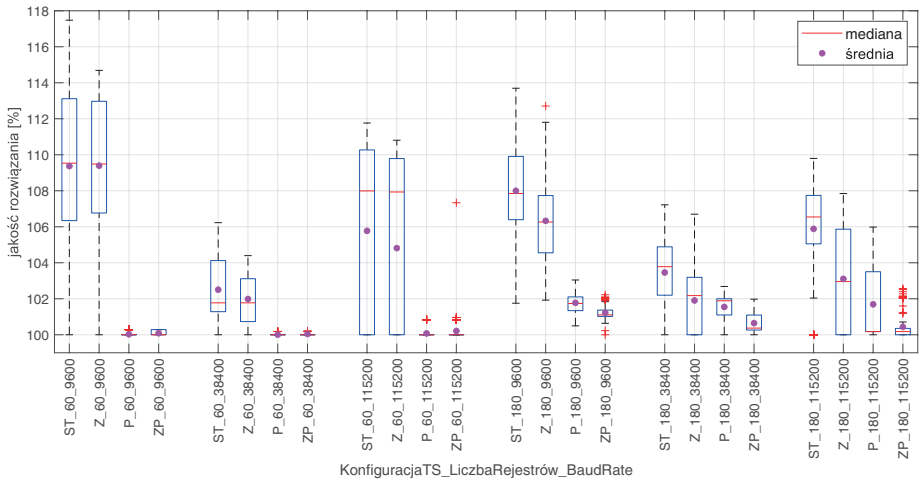
W tab. 1 podano użyte w obliczeniach wartości parametrów konfiguracyjnych algorytmu TS. Dobrano je eksperymentalnie dla uzyskania dobrej efektywności optymalizacji porównując kilkanaście wartości każdego z parametrów.

Dla każdej kombinacji prędkości transmisji i instancji testowej przeprowadzono 100 powtórzeń algorytmu po 10 minut każde. Powtarzanie jest uzasadnione ze względu na stochastyczne uwarunkowania działania algorytmu TS.

Dla zweryfikowania skuteczności rozszerzeń algorytmu TS mających poprawić jego efektywność, porównano wyniki w czterech konfiguracjach: ST – standardowa (bez rozszerzeń), Z – dodane stochastyczne zawężanie sąsiedztwa, P – dodane

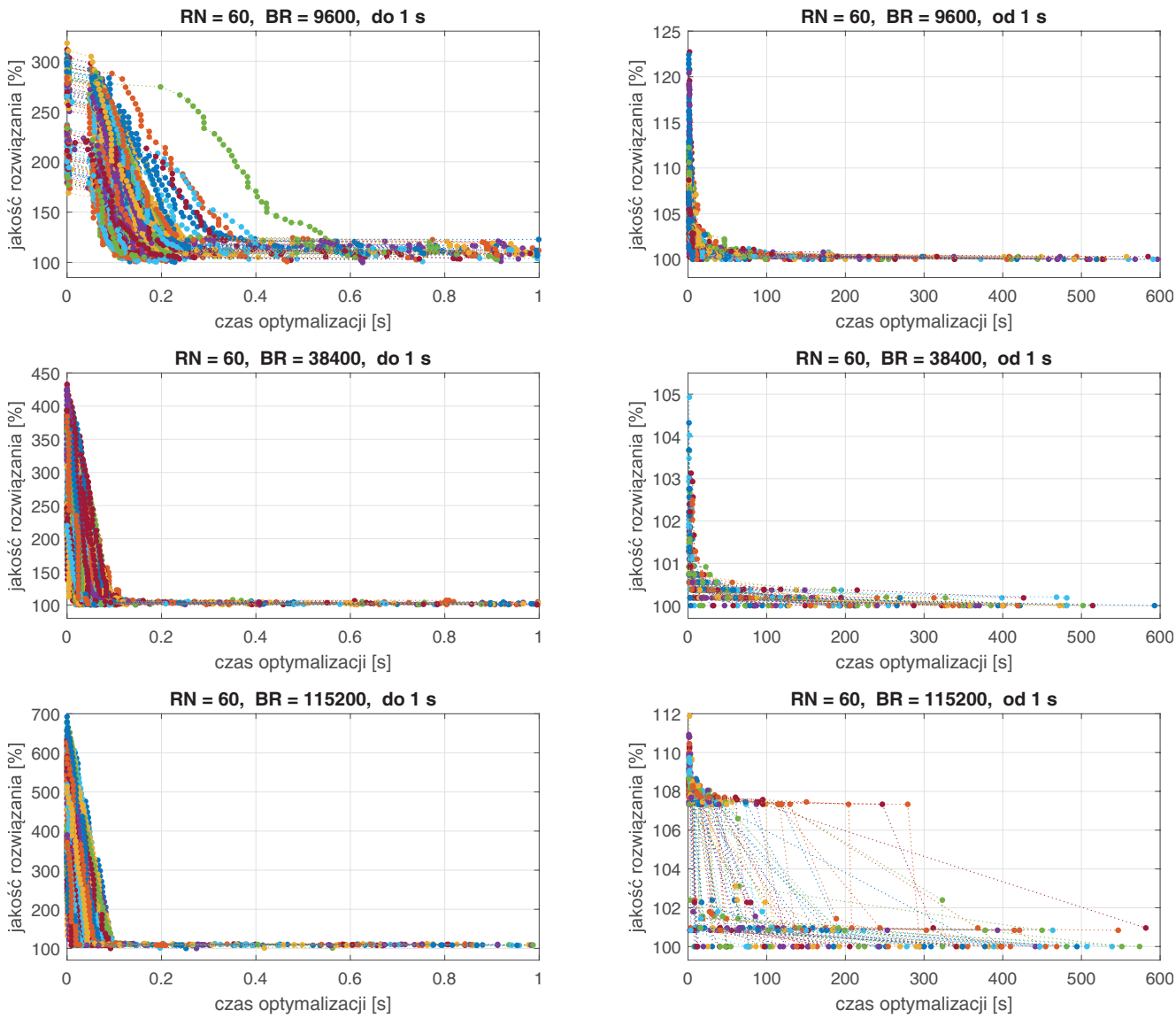
Tabela 1. Parametry konfiguracyjne algorytmu TS
Table 1. Configuration parameters of TS algorithm

Parametr	Wartość
Długość listy zakazów	1e4
Prawdopodobieństwo akceptacji p_N	0,5
Okres perturbacji D_P	1e3
Poziom perturbacji D_N	10

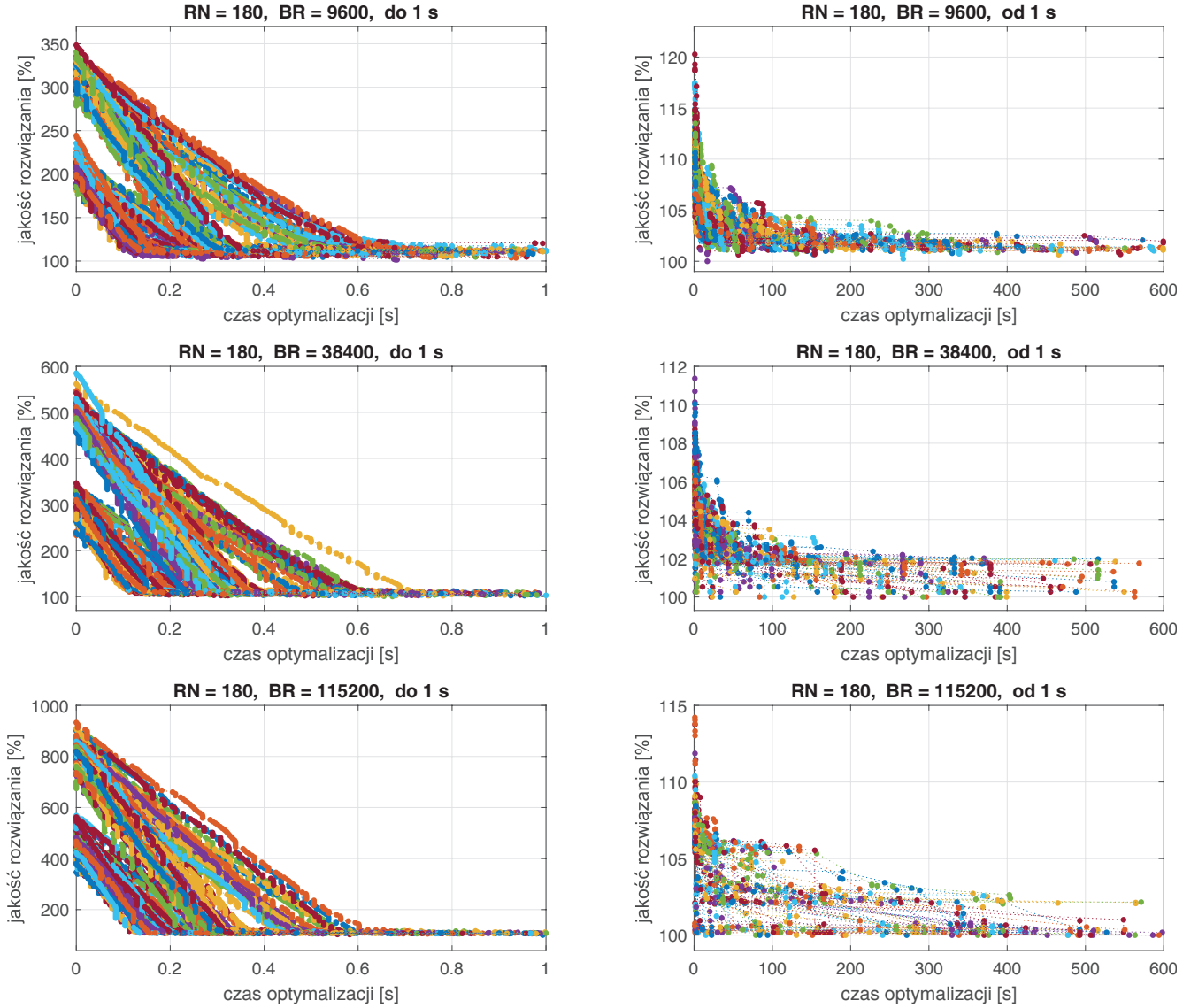


Rys. 2. Test porównawczy rozszerzeń algorytmu TS
Fig. 2. Benchmark of the TS extensions

perturbacje, ZP – jednocześnie dodane Z i P. To porównanie wykonano tylko na podstawie instancji testowej nr 1, natomiast uwzględniono warianty z liczbą rejestrów 60 oraz 180, a także 3 prędkości transmisji. Wyniki porównania są przedstawione na rys. 2 w formie wykresu pudełkowego. Wartości zostały znormalizowane względem minimum funkcji celu uzyskanego w porównywanych konfiguracjach dla danej instancji, tzn. dla kolejnych grup czterech serii danych. Wykres wskazuje, że uzupełnienie implementacji standardowej (ST) o zawężanie sąsiedztwa (Z) nieco poprawia statystyczną efektywność algorytmu dla instancji zawierającej 60 rejestrów, natomiast poprawia ją wyraźnie w przypadku 180 rejestrów. Mechanizm perturbacji



Rys. 3. Zmiana funkcji celu w czasie optymalizacji, 60 zmiennych
Fig. 3. Objective function value as a function of optimization time, 60 variables



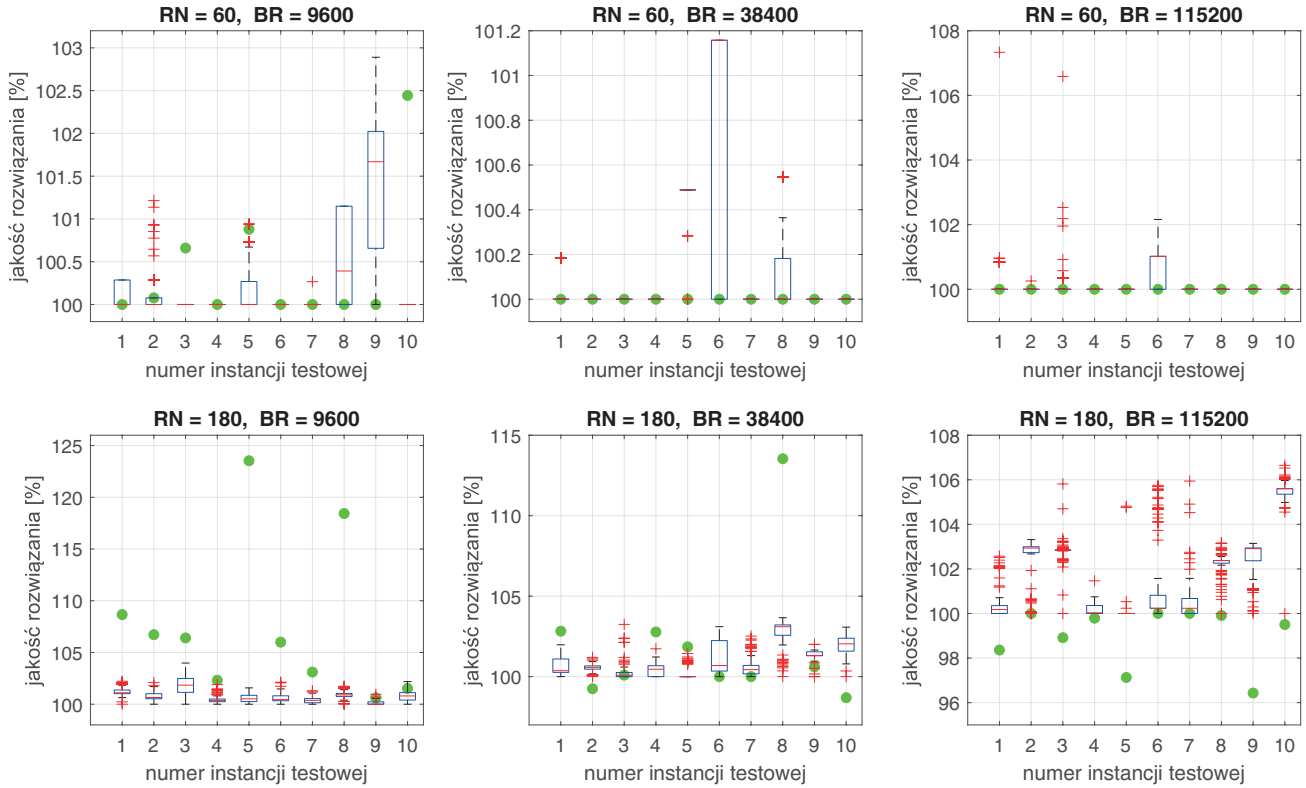
Rys. 4. Zmiana funkcji celu w czasie optymalizacji, 180 zmiennych
Fig. 4. Objective function value as a function of optimization time, 180 variables

(P) powoduje natomiast większą poprawę, bardzo dużą przy 60 rejestrach, natomiast relatywnie mniejszą, ale również znaczną w przypadku 180 rejestrów. O ile przy 60 rejestrach połączenie dwóch rozszerzeń (ZP) nie ma przewagi nad samą perturbacją, to taka przewaga pojawia się wyraźnie dla 180 rejestrów. Na podstawie wyników porównania przyjęto, że łączne zastosowanie rozszerzeń Z i P skutecznie zwiększa efektywność algorytmu, zwłaszcza dla większych instancji, i dla takiej konfiguracji algorytmu TS wykonano pełny zakres optymalizacji obejmujący wszystkie instancje.

Uzyskane przebiegi zmian funkcji celu w czasie zamieszczono na wykresach z rys. 3 i 4, odpowiednio dla 60 i 180 zmiennych. Dla zwięzłości wykresy uwzględniają tylko instancję testową nr 1 jako przykładową, charakter przebiegów dla innych instancji jest podobny. Na każdym z wykresów nałożono przebiegi ze 100 powtórzeń, a kropki oznaczają momenty i wartości poprawy wyniku. Wartości funkcji celu są zaprezentowane po przeliczeniu na ”jakość rozwiązania”, które polega na normalizacji względem najlepszego końcowego wyniku uzyskanego ze wszystkich 100 powtórzeń. Zmiany wartości funkcji celu w czasie są bardzo strome, dlatego dla czytelności podzielono je na część do 1 s (po prawej) i od 1 s (po lewej). Największa poprawa następuje bardzo szybko po uruchomieniu algorytmu, w czasie poniżej 1 s. Wiąże się to z trajektorią w przestrzeni rozwiązań pokonywaną bezpośrednio do najbliższego minimum (w sensie zdefiniowanego sąsiedztwa). Następnie algorytm dokonuje wolniejszej poprawy, w którą zaangażowane są właściwe mechanizmy przeszukiwania z zakazami.

W celu bardziej szczegółowego porównania algorytmów TS i CP opracowano wykresy z rys. 5. Wyniki TS są na nich przedstawione w postaci standardowego wykresu pudełkowego, na którym jako serie danych przyjęto końcowe wartości funkcji celu ze 100 powtórzeń dla każdej konfiguracji. Zielone znaczniki reprezentują wartości funkcji celu uzyskane przez optymalizację CP. Zestawienie wyników prowadzi do kilku wniosków:

1. Rozrzut wartości funkcji celu uzyskiwanych w odrębnych powtórzeniach TS dla tych samych danych jest niewielki, a więc wyniki są spójne i w praktycznym zastosowaniu wystarczy mała liczba uruchomień lub nawet pojedyncze. Maksymalna odchyłka od wyniku najlepszego nie przekracza 8 %, przy czym odchyłki rosną z liczbą rejestrów oraz (dla 180 zmiennych) z prędkością transmisji.
2. Dla większości testów opartych na 60 zmiennych wszystkie powtórzenia TS oraz wykonania CP dają identyczne wyniki, co pozwala sądzić, że zadania te są relatywnie łatwe, a uzyskane wartości czasu komunikacji są optymalne, nawet jeśli ta optymalność nie ma ścisłego potwierdzenia. Taka spójność wyników wskazuje także na poprawność obu algorytmów, zaimplementowanych zupełnie niezależnie i w całkowicie inny sposób.



Rys. 5. Zestawienie wyników i porównanie z metodą CP
Fig. 5. Results review and comparison to CP method

3. W przypadku największych instancji (180 zmiennych) optymalizacja CP nabiera statystycznej przewagi nad TS wraz ze wzrostem prędkości transmisji. Przyczyna tego efektu wymaga dalszych analiz, może nią być przyjęta dla TS postać sąsiedztwa. Jednakże przewaga CP nad TS nie przekracza nigdy 4 %, podczas gdy przewaga TS nad CP osiąga dla kilku instancji rząd kilkunastu procent.

Obszerny zestaw wyników eksperymentalnych z wielokrotnych powtórzeń TS pozwala wyznaczyć wiele parametrów statystycznych. Przykładowe parametry zostały przedstawione w tab. 2 i 3, odpowiednio dla 60 i 180 zmiennych, i reprezentują kolejno:

- BR – prędkość transmisji,
- Inst – numer instancji,
- Max [%] – maksymalną wartość funkcji celu TS względem wartości minimalnej ze 100 powtórzeń,
- Mean [%] – średnią wartość funkcji celu TS względem wartości minimalnej ze 100 powtórzeń,
- CV [%] – współczynnik zmienności wartości funkcji celu TS,
- BH (*best hits*) – krotność osiągnięcia wartości najlepszej (minimalnej) w 100 powtórzeniach,
- CP [%] – wartość funkcji celu CP odniesioną do najlepszej wartości funkcji celu TS,
- T [s] – czas, po którym lepszy z algorytmów dla danej instancji uzyskuje wynik korzystniejszy od końcowego wyniku drugiego z nich, jeśli wyniki są identyczne, komórka ma wartość 0,
- TSB (*TS better*) – liczba powtórzeń TS, w których algorytm ten był lepszy od CP,
- CPB (*CP better*) – liczba powtórzeń TS, w których algorytm ten był gorszy od CP,
- TS2 – czas, po którym TS uzyskał wartość mniejszą od 102 % wartości końcowej,
- CP5 – czas, po którym CP uzyskał wartość mniejszą od 105 % wartości końcowej.

Na podstawie danych z tabel można sformułować m.in. następujące wnioski

1. Parametry Max[%], Mean[%], CV[%] i BH pozwalają dokładniej ocenić rozrzut wyników przy powtarzaniu algorytmu TS. Wyraźna systematyczność wzrostu rozrzutu wyników wraz z prędkością transmisji zachodzi dla 180 zmiennych. Dla 60 zmiennych korelacja taka jest słabsza, a rozrzut mocniej zależy od konkretnej instancji, np. dla BR = 115 200 występują przypadki z zerowymi oraz bardzo dużymi rozrzutami. Aż w 16 na 30 testów dla 60 zmiennych algorytm TS uzyskuje wszystkie rozwiązania identyczne, natomiast nie ma to miejsca nigdy dla 180 zmiennych, dla których z kolei w 11 testach rozwiązanie najlepsze uzyskiwane jest tylko jednokrotnie.
2. Parametry TSB i CPB wskazują, że porównywane algorytmy są konkurencyjne w aspekcie uzyskanych wartości funkcji celu. Zależnie od konkretnej instancji zadania, TS może być we wszystkich powtórzeniach lepszy od CP, we wszystkich gorszy, w niektórych jednakowy, albo w części lepszy i w części gorszy. Szczególnie jest to widoczne dla przypadku 180 zmiennych i BR = 38 400.
3. Parametry czasowe T[s], TS2[s] i CP5[s] wyraźnie faworyzują TS pod względem szybkości optymalizacji. Gdy TS uzyskuje wynik lepszy od CP, przewaga pojawia się w czasie kilku sekund od uruchomienia (z wyjątkiem jednego przypadku), natomiast gdy przewagę ma CP, zyskuje ją nie szybciej niż po ok. 10 minutach. Ponieważ końcowe wartości funkcji celu z obu algorytmów są zwykle zbliżone, z praktycznego punktu widzenia istotniejsze staje się, po jakim czasie są one osiągane. Optymalizacja CP uzyskuje wynik 5 % powyżej optimum (parametr CP5[s]) po czasie mocno uzależnionym od konkretnej instancji, wynoszącym między 1 a 450 s (przeciętnie 10 s) dla 60 zmiennych oraz między 275 a 3160 s (przeciętnie 615 s) dla 180 zmiennych. W przypadku TS poprawa jest tak szybka, że 5 % powyżej optimum algorytm uzyskuje praktycznie natychmiast, dla-

Tabela 2. Wybrane parametry statystyczne rozwiązań, 60 zmiennych
Table 2. Selected statistical parameters of solutions, 60 variables

BR	Inst	Max [%]	Mean [%]	CV [%]	BH	CP [%]	T [s]	TSB	CPB	TS2 [s]	CP5 [s]
9600	1	0,286	0,086	0,132	70	0	0	0	30	1	450
	2	1,215	0,107	0,251	69	0,078	5	69	17	1	162
	3	0	0	0	100	0,66	1	100	0	1	8
	4	0	0	0	100	0	0	0	0	1	23
	5	0,94	0,142	0,257	71	0,879	4	98	2	1	10
	6	0	0	0	100	0	0	0	0	1	404
	7	0,267	0,003	0,027	99	0	0	0	1	1	110
	8	1,149	0,541	0,533	44	0	0	0	56	1	78
	9	2,89	1,42	0,912	18	0	0	0	82	3	42
	10	0	0	0	100	2,443	1	100	0	1	32
38 400	1	0,183	0,042	0,077	77	0	0	0	23	1	8
	2	0	0	0	100	0	0	0	0	1	12
	3	0	0	0	100	0	0	0	0	1	9
	4	0	0	0	100	0	0	0	0	1	7
	5	0,489	0,475	0,074	2	0	0	0	98	1	5
	6	1,158	0,855	0,505	26	0	0	0	74	1	10
	7	0	0	0	100	0	0	0	0	1	6
	8	0,546	0,078	0,156	74	0	0	0	26	1	15
	9	0	0	0	100	0	0	0	0	1	6
	10	0	0	0	100	0	0	0	0	1	7
115 200	1	7,333	0,211	0,785	83	0	0	0	17	1	50
	2	0,257	0,003	0,026	99	0	0	0	1	1	10
	3	6,586	0,182	0,76	84	0	0	0	16	1	15
	4	0	0	0	100	0	0	0	0	1	15
	5	0	0	0	100	0	0	0	0	1	1
	6	2,161	0,773	0,562	30	0	0	0	70	1	5
	7	0	0	0	100	0	0	0	0	1	12
	8	0	0	0	100	0	0	0	0	1	5
	9	0	0	0	100	0	0	0	0	1	5
	10	0	0	0	100	0	0	0	0	1	5

tego użyto marginesu 2 % (parametr TS2[s]), który nadal jest osiągany w większości przypadków w czasie najwyżej kilku sekund, znacznie szybciej niż 5% dla CP.

4. Czas uzyskania dobrego wyniku przez algorytm TS, którego miarą jest parametr TS2[s], zależy od konkretnej instancji, tj. rozmieszczenia transmitowanych zmiennych, jak i od szybkości transmisji. W przypadku 180 zmiennych (tab. 3) średnia wartość TS2[s] wyraźnie rośnie z prędkością transmisji, ale indywidualne czasy mocno zależą od instancji, niekiedy w sposób nieregularny, np. dla instancji nr 3 wartość TS2[s] wynosi kolejno 16, 1 i 487 s dla rosnących prędkości. Można również dokonać podziału instancji i prędkości transmisji na takie, które okazały się łatwe i prawie natychmiast uzyskano bardzo dobre rozwiązania, np. instancje nr 1, 4, 5, 6, 7 dla BR = 115 200 oraz wyraźnie trudniejsze, np. pozostałe instancje dla tej samej prędkości.

Tabela 3. Wybrane parametry statystyczne rozwiązań, 180 zmiennych
Table 3. Selected statistical parameters of solutions, 180 variables

BR	Inst	Max[%]	Mean[%]	CV[%]	BH	CP[%]	T[s]	TSB	CPB	TS2[s]	CP5[s]
9600	1	2,22	1,223	0,382	1	8,662	1	100	0	1	373
	2	2,088	0,748	0,434	5	6,723	1	100	0	1	829
	3	3,98	1,879	0,855	1	6,405	1	100	0	16	430
	4	1,906	0,451	0,324	1	2,316	1	100	0	1	2495
	5	1,581	0,557	0,348	5	23,53	1	100	0	2	617
	6	2,125	0,56	0,438	18	5,998	1	100	0	3	275
	7	1,299	0,401	0,293	1	3,109	1	100	0	3	759
	8	1,728	0,9	0,335	1	18,43	1	100	0	1	3160
	9	0,957	0,128	0,183	51	0,609	4	98	2	1	553
	10	2,187	0,789	0,476	6	1,529	9	97	3	9	347
38 400	1	1,977	0,646	0,622	23	2,824	1	100	0	1	398
	2	1,198	0,534	0,212	1	−0,754	807	0	100	1	770
	3	3,245	0,275	0,563	47	0,093	3	54	46	1	466
	4	1,727	0,431	0,386	31	2,775	1	100	0	1	570
	5	1,442	0,111	0,323	89	1,853	2	100	0	2	498
	6	3,103	1,191	1,029	18	0	0	0	82	3	502
	7	2,518	0,581	0,645	18	0	0	0	82	6	681
	8	3,66	2,719	0,772	1	13,539	1	100	0	70	654
	9	2,002	1,335	0,29	2	0,612	306	4	96	1	518
	10	3,075	1,987	0,606	1	−1,31	696	0	100	20	614
115 200	1	2,571	0,438	0,678	28	−1,639	712	0	100	1	627
	2	3,316	2,571	0,876	4	0	0	0	96	53	639
	3	5,814	2,836	0,509	1	−1,086	538	0	100	487	529
	4	1,466	0,177	0,226	50	−0,209	1146	0	100	1	656
	5	4,817	0,103	0,674	96	−2,872	565	0	100	1	565
	6	5,754	1,198	1,799	12	0	0	0	88	3	835
	7	5,943	0,602	1,013	26	0	0	0	74	3	482
	8	3,149	2,238	0,531	2	−0,089	1460	0	100	52	1118
	9	3,146	2,489	0,743	1	−3,57	659	0	100	13	659
	10	6,649	5,489	0,613	1	−0,494	662	0	100	547	643

6. Podsumowanie

Dobór odpowiedniego schematu wymian danych wpływa na wydajność komunikacji. Architektura pamięci sterowników przemysłowych implementujących pakiet inżynierski CPDev pozwala na odczyt zmiennych binarnych w protokole Modbus na różne sposoby, za pomocą dedykowanych funkcji binarnych jak też przez funkcje rejestrowe, wspólnie z innymi zmiennymi. Znalezienie optymalnego grupowania przy większej liczbie zmiennych jest złożone i wymaga zastosowania metod optymalizacyjnych.

W artykule zaproponowano użycie algorytmu przeszukiwania z zakazami w celu znalezienia grupowania skutkującego najmniejszym sumarycznym czasem cyklu komunikacyjnego. Jest to odmienne podejście niż przedstawione w [1], gdzie użyto programowania z ograniczeniami. Przeprowadzono porównanie

obu tych modeli optymalizacyjnych dla identycznych instancji testowych. W rezultacie badań eksperymentalnych można wnioskować, że:
- dla niewielkiej liczby zmiennych oba podejścia dają identyczne rezultaty, niezależnie od prędkości transmisji,
- dla dużej liczby zmiennych przy niewielkiej prędkości transmisji przeszukiwanie z zakazami daje lepsze wyniki niż programowanie z ograniczeniami, ale przy dużej prędkości transmisji *tabu-search* jest nieco gorszy.

Warto jednak zauważyć, że z praktycznego punktu widzenia niekiedy interesujące jest jak najszybsze wyznaczenie rozwiązania bliskiego optymalnemu. Tu przewagę ma przeszukiwanie z zakazami. Zazwyczaj, nawet dla dużych instancji, algorytm *tabu-search* znajduje suboptymalne rozwiązanie (nie gorsze niż 102 % końcowej wartości) w czasie poniżej 1 s, podczas gdy programowanie z ograniczeniami wymaga kilkuset sekund by znaleźć rozwiązanie mniejsze niż 105 % najlepszego. Dzięki temu możliwe jest zastosowanie przeszukiwania z zakazami do wyznaczania wstępnego rozwiązania na bieżąco, już w trakcie budowy programu, wraz z dodawaniem nowych zmiennych czy zmianą ich położenia. Co więcej algorytm *tabu-search* nie wymaga osobnego solwera jak programowanie z ograniczeniami i może być łatwo wbudowany w inne oprogramowanie. Jest to interesujący kierunek dalszych prac, możliwy do implementacji podczas rozwoju środowiska inżynierskiego CPDev. Innym potencjalnym kierunkiem może być wprowadzenie w sterowniku kopii transmitowanych zmiennych, rozmieszczonych tak, by zapewnić wydajną komunikację. Konieczne byłoby oczywiście także dodanie procesu synchronizującego zmiany wartości zmiennych, co stanowiłoby dodatkowe obciążenie. W ramach dalszych prac celowe byłoby przeprowadzenie analizy w jakich warunkach takie rozszerzenie byłoby opłacalne.

Bibliografia

1. Rzońca D., Bożek A., Optymalizacja mieszanego odczytu zmiennych binarnych i rejestrowych w protokole Modbus ze sterownika PLC implementującego CPDev, „Pomiary Automatyka Robotyka", Vol. 28, Nr 3, 2024, 83–91, DOI: 10.14313/PAR_253/83.
2. Bożek A., Rzońca D., Communication Time Optimization of Register-Based Data Transfer, "Electronics", Vol. 12, No. 24, 2023, DOI: 10.3390/electronics12244917.
3. Glover F., Tabu Search—Part I, "ORSA Journal on Computing", Vol. 1, No. 3, 1989, 190–206, DOI: 10.1287/ijoc.1.3.190.
4. IEC, IEC 61131-3 – Programmable controllers – Part 3: Programming languages, 2003, 2013.
5. Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L., *Developing a Multiplatform Control Environment*, "Journal of Automation, Mobile Robotics and Intelligent Systems", Vol. 13, No. 4, 2019, 73–84, DOI: 10.14313/JAMRIS/4-2019/40.
6. Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L., *CPDev engineering environment for control programming*, [In:] Trends in Advanced Intelligent Control, Optimization and Automation, AISC, Vol. 577, 2017, 303–314, DOI: 10.1007/978-3-319-60699-6_29.
7. Hubacz M., Sadolewski J., Trybus B., *Model i implementacja dwurdzeniowego sterownika programowalnego opartego na maszynie wirtualnej*, „Pomiary Automatyka Robotyka", Vol. 28, Nr 3, 2024, 93–99, DOI: 10.14313/PAR_253/93.
8. Hajduk Z., Trybus B., Sadolewski J., *Architecture of FPGA Embedded Multiprocessor Programmable Controller*, "IEEE Transactions on Industrial Electronics", Vol. 62, No. 5, 2015, 2952–2961, DOI: 10.1109/TIE.2014.2362888.
9. Trybus B., *Development and Implementation of IEC 61131-3 Virtual Machine*, "Theoretical and Applied Informatics", Vol. 23, No. 1, 2011, 21–35, DOI: 10.2478/v10179-011-0002-z.
10. Sadolewski J. Trybus B., *Compiler and virtual machine of a multiplatform control environment*, "Bulletin of the Polish Academy of Sciences. Technical Sciences", Vol. 70, No. 2, 2022, DOI: 10.24425/bpasts.2022.140554.
11. Kwiecień R., *Komputerowe systemy automatyki przemysłowej*, Helion, 2012.
12. Stój J., *Wybrane zagadnienia sieci komunikacyjnych w przemysłowych systemach komputerowych*, Wydawnictwo Politechniki Śląskiej, 2023.
13. IEC, IEC 61158 – Industrial Communication Networks – Fieldbus Specifications, 2007.
14. Gaj P., Jasperneite J., Felser M., *Computer Communication Within Industrial Distributed Environment – a Survey*, "IEEE Transactions on Industrial Informatics", Vol. 9, No. 1, 2013, 182–189, DOI: 10.1109/TII.2012.2209668.
15. Scanzio S., Wisniewski L., Gaj P., *Heterogeneous and dependable networks in industry – A survey*, "Computers in Industry", Vol. 125, 2021, DOI: 10.1016/j.compind.2020.103388.
16. Titaev A., *Reducing update data time for exchange via MODBUS TCP protocol by controlling a frame length*, "Automatic Control and Computer Sciences", Vol. 51, 2017, 357–365, DOI: 10.3103/S014641161705008X.
17. Găitan V.G., Zagan I., *Modbus Protocol Performance Analysis in a Variable Configuration of the Physical Fieldbus Architecture*, "IEEE Access", Vol. 10, 2022, 123942–123955, DOI: 10.1109/ACCESS.2022.3224720.
18. Zagan I., Găitan V.G., *Enhancing the Modbus Communication Protocol to Minimize Acquisition Times Based on an STM32-Embedded Device*, "Mathematics", Vol. 10, No. 24, 2022, DOI: 10.3390/math10244686.
19. Găitan V.G., Zagan I., *Experimental Implementation and Performance Evaluation of an IoT Access Gateway for the Modbus Extension*, "Sensors", Vol. 21, No. 1, 2021, DOI: 10.3390/s21010246.
20. Găitan N.C., Zagan I., Găitan V.G., *Proposed Modbus Extension Protocol and Real-Time Communication Timing Requirements for Distributed Embedded Systems*, "Technologies", Vol. 12, No. 10, 2024, DOI: 10.3390/technologies12100187.
21. Bednarek M., Będkowski L., Dąbrowski T., Wybrane funkcje systemu dozorująco-terapeutycznego w układzie komunikacji, „Diagnostyka", Vol. 34, 2005, 31–36.
22. Zhou H., Liu H., Lv C., Zhang C., Shen W., *A path relinking with tabu search algorithm for solving hybrid flow shop scheduling problem considering multiple critical paths*, "Computers & Operations Research", Vol. 170, 2024, DOI: 10.1016/j.cor.2024.106783.
23. Saida E.B., Hamid O., Fouad G., Saad G., Nada M., Ismael J., *Tabu Search Algorithm for Optimal Electric Water Heating Energy Management*, "IFAC-PapersOnLine", Vol. 58, No. 13, 2024, 835–840, DOI: 10.1016/j.ifacol.2024.07.586.
24. Yamamoto R.Y., Pinto T., Romero R., Macedo L.H., *Specialized tabu search algorithm applied to the reconfiguration of radial distribution systems*, "International Journal of Electrical Power & Energy Systems", Vol. 162, 2024, DOI: 10.1016/j.ijepes.2024.110258.
25. Peng B., Wang S., Liu D., Su Z., Lü Z., Glover F., *Solving the incremental graph drawing problem by multiple neighborhood solution-based tabu search algorithm*, "Expert Systems with Applications", Vol. 237, Part A, 2024, DOI: 10.1016/j.eswa.2023.121477.
26. Rzońca D., *Poprawa wydajności komunikacji sterownika przemysłowego z panelem operatorskim HMI w środowisku*

inżynierskim CPDev, „Pomiary Automatyka Robotyka”, Vol. 24, No. 1, 2020, 35–40, DOI: 10.14313/PAR_235/35.
27. Rzońca D., *Przyspieszenie wymiany danych w protokole Modbus między PLC a HMI wykorzystującymi pakiet inżynierski CPDev*, „Pomiary Automatyka Robotyka”, Vol. 26, No. 4, 2022, 85–89, DOI: 10.14313/PAR_246/85.
28. Hubacz M., Sadolewski J., Trybus B., *Obsługa typów danych normy PN-EN 61131-3 w architekturze ARM z ograniczeniami dostępu do pamięci*, „Pomiary Automatyka Robotyka”, Vol. 26, No. 1, 2022, 23–31, DOI: 10.14313/PAR_243/23.

Tabu Search Algorithm for Communication Time Optimization in the CPDev Engineering Environment

Abstract: Efficient communication is crucial for the correct operation of distributed control systems. This article is focused on the problem of reducing the communication cycle time during reading mixed register-binary data in the Modbus protocol from an industrial controller, especially implementing the CPDev engineering environment. A tabu-search algorithm was designed and implemented for optimization. Extensive computational experiments were performed and their results were compared with the results of previous studies in which constraint programming was used for the same problem. Comparison of the results revealed that the new algorithm is much faster and therefore suitable for practical use in real-time Modbus communication planning for the CPDev environment.

Keywords: tabu search, industrial controller, PLC, communication, Modbus, CPDev

dr inż. Dariusz Rzońca
drzonca@kia.prz.edu.pl
ORCID: 0000-0001-5724-0978

Licencjat matematyki (Uniwersytet Rzeszowski 2002), magister inżynier informatyki (Politechnika Rzeszowska 2004), doktor nauk technicznych w dyscyplinie informatyka, specjalność przemysłowe systemy informatyki (Politechnika Śląska 2012). Od 2004 roku asystent, a od 2013 adiunkt w Katedrze Informatyki i Automatyki Politechniki Rzeszowskiej. Jego zainteresowania naukowe koncentrują się na kolorowanych sieciach Petriego oraz zagadnieniach związanych z komunikacją w systemach automatyki.

dr inż. Andrzej Bożek
abozek@prz.edu.pl
ORCID: 0000-0003-3015-7474

Absolwent Wydziału Elektrotechniki i Informatyki Politechniki Rzeszowskiej (2008). Stopień doktora nauk technicznych w dyscyplinie informatyka uzyskał w 2015 r. Pracuje jako adiunkt w Katedrze Informatyki i Automatyki Politechniki Rzeszowskiej. Jego zainteresowania naukowe dotyczą optymalizacji dyskretnej, harmonogramowania zadań oraz projektowania układów sterowania.