Implementation and Complexity Comparison of Optimal Power Allocation Algorithms in Distributed Passive Radar Systems

Omid Taghizadeh and Gholamreza Alirezaei
{taghizadeh, alirezaei}@ti.rwth-aachen.de

Abstract—In [1], an optimal algorithm for sensor selection in passive sensor networks is derived. Based on that algorithm, we first present its MATLAB implementation and subsequently we propose an alternative implementation. The alternative implementation, called Algorithm 1 hereinafter, includes a sorting mechanism which reduces the number of iterations and the computational load simultaneously. A detailed investigation of Algorithm 1 along with its comparison to the one in [1] is submitted for publication to an international conference.

alpha = abs(g)/sqrt(M);
beta = sqrt(N*(R*abs(g).^2+M)/(M*abs(h).^2));
Reliability = beta/alpha;
alphabeta = beta.*alpha;
theta = -angle(g.*h);
FK = 1:K;
Kmax = [ ];
Kp = [1];
Prem = P_tot-K*0;
IterCnt_ref1 = 0;
while ~isempty(Kp) && (length(Kmax)<K)
    Klin = setdiff(FK, Kmax);
    Kn = [1];
    while ~isempty(Kn) && ~isempty(Klin)
        iteration_cnt_reference_1 = iteration_cnt_reference_1+1;
        chi = (Prem + sum(betav(Klin).^2 + 0))/sum(alphabeta(Klin));
        Xk = alphabeta(Klin).*chi - Reliability(Klin));
        Kn = Klin(find(Xk <= 0));
    end

Both authors are with the Institute for Theoretical Information Technology, RWTH Aachen University, D-52074 Aachen, Germany.
Algorithm 2

\[ F_0 = [] \]
\[ F_{\text{saturated}} = [] \]
\[ F_{\text{active}} = [] \]

% calculate derivations to constitute the sequence B:
\[ \text{der}_0 = \text{alpha}_2 \text{beta}_2 / ((0 + \beta_1)^2) \]
\[ \text{der}_\text{P_max} = \text{alpha}_2 \text{beta}_2 / ((\text{P_max} + \beta_1)^2) \]
\[ [B, I] = \text{sort}([\text{der}_0; \text{der}_\text{P_max}], 1, \text{'ascend'}) \]

% bi-section search on the incremental regions in B
\[ b_{\text{max}} = 2^*K_1 + 1 \]
\[ b_{\text{min}} = 0 \]
iteration_cnt_algorithm_1 = 0; \% counting the iterations
while (1)

\[ \text{iteration_cnt_algorithm_1} = \text{iteration_cnt_algorithm_1} + 1 \]
\[ \text{bisection_idx} = \text{floor}((b_{\text{max}} + b_{\text{min}})/2); \]
\[ \text{ref1} = \text{zeros}(2^*K_1) \]
\[ \text{ref1}((\text{bisection_idx} + 1):2^*K_1, 1) = \text{ref1}((\text{bisection_idx} + 1):2^*K_1, 1); \]
\[ \text{ref2} = \text{ref1}(1:K_1) + \text{ref1}(K_1:2^*K_1, 1) \]
\[ F_0 = \text{find} (\text{ref2} == 0); \]
\[ F_{\text{saturated}} = \text{find} (\text{ref2} == 2); \]
\[ F_{\text{active}} = \text{find} (\text{ref2} == 1); \]

if (length(F_{\text{active}}) == 0)
\[ \text{temp1} = P_{\text{max}} * \text{length}(F_{\text{saturated}}) - \text{min}(P_{\text{tot}}, K_{\text{P_max}}) \]
if (temp1 == 0)
\[ \text{break}; \]
elseif(temp1 > 0)
\[ \text{b_min} = \text{bisection_idx}; \]
continue;
elseif(temp1 < 0)
\[ \text{b_max} = \text{bisection_idx}; \]
continue;
end

end

WaterLevel = ((\text{sum}(\text{alphabet}(F_{\text{active}}, 1)))/(\text{P_tot} - \text{length}(F_{\text{saturated}})*P_{\text{max}} + \text{sum}(\text{beta}_2(F_{\text{active}}, 1))))^2 \]

if (\text{min}(P_{\text{tot}}, K_{\text{P_max}}) < \text{length}(F_{\text{saturated}}) \times P_{\text{max}})
\[ \text{b_min} = \text{bisection_idx}; \]
continue;
elseif (\text{WaterLevel} > B(\text{bisection_idx} + 1, 1))
\[ \text{b_min} = \text{bisection_idx}; \]
continue;
elseif (\text{WaterLevel} < B(\text{bisection_idx}, 1))
\[ \text{b_max} = \text{bisection_idx}; \]
continue;
else
\[ \text{break}; \]
end

deltatime_algorithm_1 = deltatime_algorithm_1 + toc;

\% Proposed algorithm using sorting mechanism (Algorithm 1)
\% achieved power allocation scheme:
\% Xk = alphabeta.*($(\text{chi} - \text{Reliability})$);
\% Xk(Kmin) = 0;
\% Xk(Kmax) = P_{\text{max}};

\% achieved power allocation scheme:
\% Xk = alphabeta.*($(\text{chi} - \text{Reliability})$);
\% Xk(Kmin) = 0;
\% Xk(Kmax) = P_{\text{max}};

\% achieved power allocation scheme:
\% Xk = alphabeta.*($(\text{chi} - \text{Reliability})$);
\% Xk(Kmin) = 0;
\% Xk(Kmax) = P_{\text{max}};

\% achieved power allocation scheme:
\% Xk = alphabeta.*($(\text{chi} - \text{Reliability})$);
\% Xk(Kmin) = 0;
\% Xk(Kmax) = P_{\text{max}};
WaterLevel)) - beta_2(F_active,1);
Xk_bisection (F_0,1) = 0;
Xk_bisection (F_saturated,1) = P_max;
end

% comparison metrics
AVG_ellapced_time_reference1 = ellapced_time_reference1/NInstance;
AVG_ellapced_time_alg1 = ellapced_time_alg1/NInstance;
AVG_iteration_cnt_algorithm_1 = iteration_cnt_algorithm_1/NInstance;
AVG_iteration_cnt_reference_1 = iteration_cnt_reference_1/NInstance;

REFERENCES